Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de un dashboard para la monitorización de la actividad de asociaciones musicales

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Roberto Lendínez, Jesús Yoel

Tutor: Vidal Oriola, Germán Francisco

2020-2021

Resumen

En este trabajo se ha planteado una solución para líderes o administradores de las federaciones de sociedades musicales, cuya necesidad es acceder a los datos para monitorizar y gestionar los datos de las sociedades que se encuentran bajo su dominio. Este panel de control permite a los líderes de federaciones consultar un listado de sus sociedades musicales registradas, datos y detalles sobre las mismas y monitorizar datos, estadísticas y eventos de la propia federación de un modo sencillo y accesible desde cualquier navegador web en cualquier dispositivo utilizando las últimas tecnologías.

Palabras clave: federación, sociedad, música, React, proyecto, panel de control, consulta.

Abstract

This work presented a solution for leaders or administrators of the federations of musical partnerships, who need to access the data to manage the data of the partnerships that are under their domain. This dashboard allows federation leaders to consult a list of their registered musical partnerships, data and details about them and monitor data, statistics and events of the federation itself in a simple and accessible way from any web browser in any device using the latest technologies.

Keywords: federation, partnership, music, React, project, dashboard, consult

Tabla de contenidos

1. Introducción

Ésta es la memoria del proyecto en el que se basa el Trabajo Final de Grado en Ingeniería Informática cursado en la Escuela Técnica Superior de Ingeniería Informática de la Universidad Politécnica de Valencia. A lo largo del trabajo veremos la evolución y desarrollo del proyecto, desde su planteamiento en la empresa que se ha realizado, hasta su implementación, integración y despliegue en la página web de la empresa en la que se ha realizado el proyecto en toda su plenitud.

La idea de la cual proviene este proyecto nació mucho antes de que el autor de esta memoria fuera contratado por la *start-up* Glissandoo, empresa en la que se ha realizado el proyecto. Esta idea nació de la mano del CEO y CTO de la empresa y, más tarde, una vez el autor se encontraba realizando las prácticas en la empresa, serviría como planteamiento de un proyecto para el Trabajo Final de Grado.

La idea en la cual se basa el proyecto, es decir, crear un panel de control para monitorizar las actividades y estadísticas de las sociedades de federaciones musicales, nace de la necesidad de organizar, centralizar, unificar y ofrecer para consulta todas las fuentes individuales con las que cuentan las federaciones (grupos en la aplicación de mensajería WhatsApp; correos electrónicos; documentos y hojas de cálculo subidos en la nube, ya sea en Google Drive, Dropbox, OneDrive, etc.) en una sola herramienta, que proporcione todos esos datos en una única localización: el propio panel de control.

Para esto, las propias sociedades musicales de las federaciones tienen que dar su consentimiento para que tanto la empresa Glissandoo como la federación musical a la que pertenece esa sociedad puedan tratar los datos de forma legal (por la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales).

Para la elaboración de este proyecto se hará uso de las últimas novedades tecnológicas para el desarrollo de aplicaciones web, para favorecer el rendimiento, la accesibilidad, las buenas prácticas y el posicionamiento SEO, así como la integridad de los datos mediante el uso de servidores online.

1.1 Contexto

Las federaciones de sociedades musicales son organizaciones que se encargan de gestionar las actividades, eventos, material, logística y otros tipos de necesidades que tienen las sociedades musicales. Para ello, las sociedades musicales tienen que tener constancia, así como un registro, de todo lo anterior.

Sin embargo, como se ha mencionado en la introducción, muchas de las federaciones musicales (al menos, de las que se ha tenido constancia en la búsqueda de mercado en España) usan métodos poco organizados o hasta informales para llevar el registro de las actividades, eventos y datos de los miembros de las sociedades. Además, los métodos de comunicación entre miembros de las federaciones y sociedades y, en ocasiones, la transmisión de este tipo de datos en documentos como archivos en formato PDF, Word u hoja de cálculo (Excel), se lleva a cabo mediante aplicaciones de mensajería rápida. Más concretamente, por grupos de WhatsApp, y esto da como resultado la falta de constancia o registro oficial en métodos de comunicación más formales.

De esta forma, mucha de la información está almacenada de forma incorrecta o poco eficiente, dándose casos en los que se gasta mucho tiempo simplemente tratando de encontrar esta información (documentos, conversaciones, acuerdos, etc.) o incluso llega a perderse información valiosa o importante.

1.2 Motivación

Para encontrar soluciones a los problemas mencionados en el apartado anterior, nació la empresa Glissandoo, y con ella uno de sus proyectos con más potencial: un panel de control que permita a los líderes de las federaciones musicales consultar los datos más importantes, las estadísticas y los números de cada sociedad dentro de su dominio. Este panel tiene como principal función poner al alcance de cualquier líder de federación musical los datos de cada una de las sociedades musicales que se encuentran en su dominio.

Esta funcionalidad siempre vendrá condicionada por la voluntad de las sociedades musicales, que podrán elegir dar permiso o negarlo para ceder estos datos

a Glissandoo, de forma que la empresa pueda mostrarlos y proporcionarlos a los líderes de las federaciones.

Como alumno en prácticas en la *start-up* Glissandoo, se me dio la oportunidad de trabajar en solitario (aunque con la ocasional ayuda del CTO de la empresa) en este proyecto que desde un principio me llamó la atención. La idea a largo plazo de este panel de control es que pueda llegar a ser una herramienta usada por un miembro del senado que supervisa la actividad de las federaciones musicales. Este miembro podrá aportar en sus funciones oficiales datos y estadísticas de las federaciones al senado, estadísticas y números que podrá consultar en el panel de control de Federaciones de Glissandoo con unos permisos especiales.

1.3 Objetivos del proyecto

La aplicación web a desarrollar, a saber, un panel de control de monitorización de actividades y estadísticas de sociedades musicales, tiene como objetivo ser una herramienta de consulta, en la que los líderes de las federaciones musicales puedan tener al alcance, así como recopilar, mostrar y tomar conclusiones de los datos proporcionados por las sociedades musicales. Entre ellos se darán a conocer, si esa es la voluntad de las sociedades musicales, el nombre, el teléfono, el correo electrónico, etc., y algunas estadísticas de las sociedades bajo el dominio de la federación musical a la que pertenezca el líder.

La idea básica del panel de control es una página web cuya principal funcionalidad es proporcionar información sobre las sociedades que están bajo el dominio de la federación. Lo primero que verá el usuario que acceda al panel de control (que por lo general un líder de federación o similar) será la lista de sociedades musicales perteneciente a la federación a la que pertenezca, además de algunas estadísticas tanto de la federación como de las sociedades (estas últimas dentro de las mismas sociedades). Una vez dentro de cada sociedad, además de algunas estadísticas más ya mencionadas, se mostrarán algunos datos de la sociedad, y dependiendo de si la sociedad ha decidido compartir los datos con la federación y con Glissandoo, se mostrarán datos adicionales sobre la sociedad musical en cuestión. Todo esto, de nuevo, son datos que podrá consultar el líder de la federación de sociedades musicales.

También se ofrece la posibilidad de que el líder gestione los datos de la federación, por ejemplo, el nombre de la misma, su dirección, su correo electrónico, su teléfono y algunos datos más. Por último, y por ahora, se dará la posibilidad de generar un enlace que se enviará por correo a las sociedades que no se hayan “conectado” con su federación en Glissandoo para acceder a una página pública donde se pedirá consentimiento para “conectar”. Conectar en este contexto significa que la sociedad ceda sus datos para que la federación y Glissandoo puedan tener acceso a ellos, y Glissandoo pueda mostrarlos a la federación a la que pertenece la sociedad.

El objetivo de este panel de control no es proporcionar un medio de comunicación para transmitir los datos de las sociedades almacenados en el mismo panel, sino simplemente **dar acceso a los datos para su consulta**. En la aplicación web Glissandoo ya existe un chat para miembros de las sociedades y bandas, con el fin de ofrecer un canal centralizado en el cual pueda haber comunicación específicamente para organizar y gestionar asuntos relacionados con las sociedades y federaciones. Pero este chat se encuentra en un panel de control distinto y aislado del panel de control que se está describiendo en este proyecto.

Por tanto, como principales objetivos de la aplicación se destacan los siguientes:

- Permitir el acceso al panel de control sólo a líderes de federaciones autorizadas.
- Permitir la modificación de los datos de las federaciones musicales, como por ejemplo personalizar el nombre, el logo, etc.
- Proporcionar listas con las sociedades musicales pertenecientes al dominio de las federaciones registradas.
- Proporcionar datos sobre las sociedades musicales bajo la autoridad de las federaciones registradas.
- Generar estadísticas sobre los datos de las federaciones mediante los datos de las sociedades musicales bajo su dominio.
- Hacer disponible información de próximos eventos, o eventos ya realizados.
- Mostrar gráfica de número de eventos al mes del último año.
- Interactuar con un calendario con los días de eventos programados.
- Permitir el almacenamiento de estos datos en una sola ubicación centralizada, sin que las federaciones se hallen con la necesidad de solicitarlos directamente a las sociedades musicales.
- Facilitar a las federaciones musicales la solicitud de cesión de datos a las sociedades musicales que se hallan bajo su autoridad

- Estar disponible de forma online (en cualquier navegador de internet) y permitir accesibilidad desde cualquier tipo de dispositivo.
- Como objetivo adicional, se procurará hacer un uso acertado, correcto y eficiente de las nuevas tecnologías y librerías que proporciona el lenguaje de programación web JavaScript, sobre todo en la parte *front-end*.

1.4 Estructura de la memoria

La memoria se encuentra dividida en varios capítulos, recopilando toda la información relacionada con el desarrollo de este proyecto. A continuación, se proporciona un resumen del contenido de cada uno de estos capítulos.

CAPÍTULO 1- INTRODUCCIÓN

En este capítulo se establece el contexto que alberga el desarrollo del proyecto, describiendo escuetamente las ideas de éste y los objetivos que se pretenden alcanzar, presentando las motivaciones y soluciones para resolver el problema planteado.

CAPÍTULO 2- ESTADO DEL ARTE

En este capítulo veremos algunas metodologías de desarrollo usadas en el desarrollo del proyecto, un contexto de qué entornos y organizaciones musicales se están tratando, se verán aplicaciones similares a Glissandoo y porqué es necesario una nueva aplicación web que complete las opciones existentes, y las tecnologías de *front-end* y *back-end* que se han usado en este proyecto y por qué se han elegido en específico frente a otras.

CAPÍTULO 3 - ELICITACIÓN DE REQUISITOS

En este capítulo se agrupan todos los requisitos funcionales que deberá tener el proyecto al llegar al final de su desarrollo.

CAPÍTULO 4 - ANÁLISIS DEL SISTEMA

El Análisis del sistema describe la estructura y funcionalidades de la aplicación desarrollada mediante diagramas que faciliten una mejor comprensión del sistema. En este apartado se incluyen diagramas de clases UML y diagramas de casos de uso. que describen los principales comportamientos del panel de control a través de los diferentes actores.

CAPÍTULO 5 - DISEÑO E IMPLEMENTACIÓN

Este capítulo está dedicado a la fase de diseño e implementación del proyecto. Tomando como eje central las necesidades y la información de las anteriores etapas de requisitos y análisis, se crean modelos de arquitectura de alto nivel que detallan los diferentes componentes que forman el panel de control, con el fin de su futura implementación.

Se ha decidido recoger en este apartado los aspectos más importantes del desarrollo del proyecto, haciendo referencia a las tecnologías utilizadas y los aspectos más relevantes de la implementación.

CAPÍTULO 6 - RESULTADOS, CONCLUSIONES Y FUTURO

Este capítulo recoge las conclusiones extraídas durante todo el proceso de desarrollo del proyecto en función de los objetivos iniciales y los resultados obtenidos. A su vez, describe los conocimientos adquiridos durante el proceso de desarrollo y algunos aspectos que deben ser considerados en el futuro.

CAPÍTULO 7 - REFERENCIAS BIBLIOGRÁFICAS

En este capítulo se detalla la bibliografía consultada durante el desarrollo del proyecto y/o la memoria, y algunas referencias en las que se puede encontrar más información del tema del que se esté hablando.

CAPÍTULO 8 - ANEXOS

En los anexos se incluye documentación externa y una explicación más amplia de algunos aspectos mencionados en el proyecto.

2. Estado del arte

El objetivo de este capítulo de la memoria es presentar aquellas metodologías de trabajo y tecnologías de desarrollo empleadas en el presente proyecto. La finalidad que encierran los siguientes apartados es facilitar al lector la comprensión de los capítulos posteriores y esclarecer todos los conceptos que se usen en el proyecto, de forma que pueda valorar en mejores condiciones las contribuciones realizadas al proyecto.

El producto desarrollado se ha realizado usando varias metodologías de desarrollo, por lo que se explicará con más detalle qué metodologías se ha usado a lo largo del proyecto, por qué se han escogido estas metodologías, y qué partes específicas de cada una se han empleado en conformar la metodología final que se ha seguido en este proyecto.

Además, se aborda el estudio y evaluación de diversas herramientas y tecnologías existentes, en qué han beneficiado al autor del proyecto y al propio proyecto, y el porqué de su utilización en el desarrollo de la aplicación web.

2.1 Metodologías del Proceso de Software

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software informático. El objetivo de las distintas metodologías es el de intentar organizar los equipos de trabajo para que estos desarrollen las funciones de un programa de la mejor manera posible.

Regularmente este tipo de metodología tiene la necesidad de venir documentadas para que los programadores que estarán dentro de la planificación del proyecto comprendan perfectamente la metodología y, en algunos casos, el ciclo de vida del software que se pretende seguir.

Se pueden diferenciar dos tipos o corrientes diferentes de metodologías de desarrollo principales: las metodologías tradicionales y las metodologías ágiles.

Este proyecto ha estado orientado desde un principio al uso de metodologías ágiles. En concreto, se han empleado aspectos de cuatro metodologías de desarrollo de software a lo largo del proyecto. En el anexo se explica en qué consisten las metodologías empleadas. La Figura 1 nos muestra el esquema en el que se basará la elección de estas metodologías:

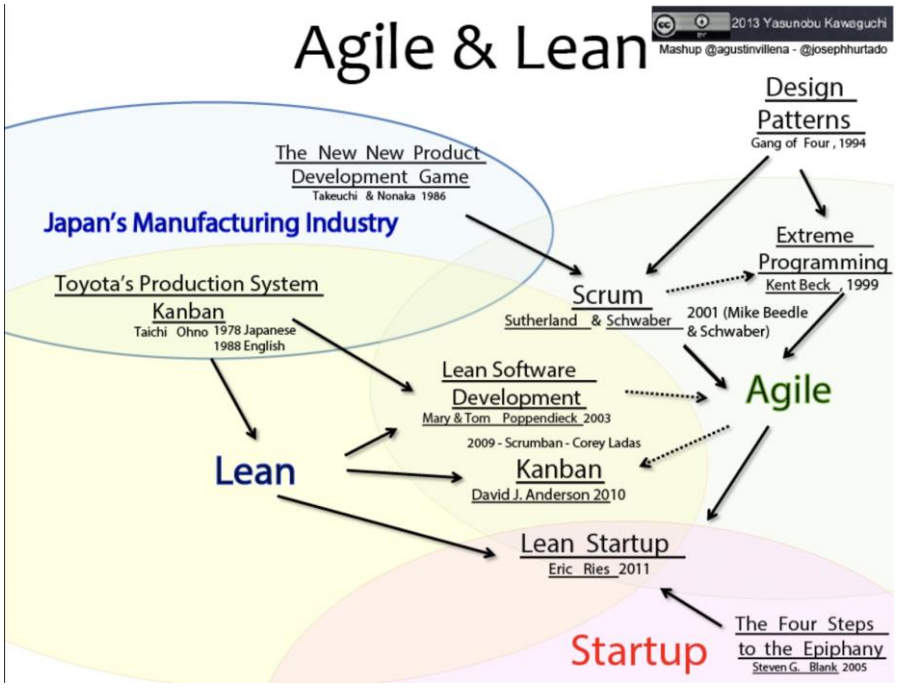


Figura 1 - Esquema de metodologías ágiles y su relación con Lean

En primer lugar, cabe mencionar que, al haber realizado las prácticas en una Startup, es lógico pensar que se ha seguido su metodología de desarrollo. Por tanto, unas de las metodologías en las que nos hemos basado para realizar el proyecto ha sido la de Startup.

Otra metodología que se ha puesto en práctica ha sido la *Extreme Programming.* En este caso, esta metodología se ha aplicado mediante la comunicación (*feedback*) que nos proporcionaban los clientes. Estos, mediante diferentes tipos de mensajes y correos, nos hacían llegar informes de errores, sugerencias para añadir o cambiar funcionalidades, o pedían ayuda en caso de necesitar información para un correcto uso de la aplicación.

Además, en lo que respecta al *Extreme Programming,* se puso en práctica también mediante el equipo de desarrollo, ya que en alguna ocasión se llevó a cabo alguna sesión de programación en pareja para resolver conflictos y dificultades que

requieran la ayuda del CTO de la empresa, principal conocedor de la estructura y modelos dentro de la aplicación principal y de la base de datos.

Una tercera metodología usada fue la metodología Kanban. Esta se llevó a cabo gracias a la sección de Proyectos de GitHub. En esta sección, puede usarse un tablero Kanban donde, a medida que iba avanzando el proyecto, iban surgiendo, evolucionando, llevándose a implementación o finalizando las diferentes tareas asignadas al proyecto, y así lograr una mejor estructuración de trabajo. Esto permitió al alumno dar prioridad a aquellas tareas que viera conveniente para la correcta evolución del trabajo, y tener en cuenta las tareas pendientes.

Por último, se ha seguido la metodología SCRUM mediante reuniones semanales. Cada lunes por la mañana se hacía una reunión en la que se aclaraban y registraban los objetivos a trabajar durante la semana. Y cada viernes, justo al final de la jornada laboral, se hacía otra reunión para explicar el progreso de los objetivos mencionados el lunes, y hacer retrospectiva del trabajo realizado. En gran parte de las reuniones de final de semana, se realizaban demostraciones sobre el producto en su respectivo estado de desarrollo, por lo que esto motivó a desarrollar el panel de control usando la estrategia del MVP (*Minimum Viable Product* o Producto Viable Mínimo).

Se usaron *sprints* semanales para definir los objetivos, sin embargo, a largo plazo se definieron ciertos plazos para cumplir los objetivos. Las dos primeras semanas tenían que dedicarse a la instalación de todas las herramientas del proyecto, además de definir la estructura de la interfaz y preparar las funcionalidades principales y transición entre páginas. El siguiente mes se dedicaría a conectar a la nube todo el panel de control y asegurar su funcionamiento usando la base de datos de Firebase. Más adelante, las siguientes dos semanas se dedicaría a revisar estilos, refactorizar código y dejar el panel listo para su uso más inmediato. Después de algunas demostraciones, decidieron añadirse algunas funcionalidades más, como mostrar la sección de Eventos. Este proceso, junto con mejorar el *responsive* (adecuación al tamaño de pantalla) de la aplicación, conllevan otras dos semanas. Las últimas semanas el equipo se dedicó a la puesta a punto del panel para futuras demostraciones con clientes, y se comenzó un proceso de *testing* que se extenderá en el futuro.

2.2 Organizaciones musicales

Para entender a plenitud este proyecto, es necesario entender cuál es el público al cual está dirigido. Se hará referencia a aquellas organizaciones con las cuáles se tendrá trato, se ha tenido trato o que formarán parte de alguna manera en el proyecto.

El software desarrollado está dirigido a todas estas organizaciones, con el objetivo de facilitar una gestión más organizada, unificada y consiguiendo que necesiten depender de los mínimos asuntos posibles mediante otro medio que no sea el software presentado en este proyecto.

2.2.1 Confederación Española de Sociedades Musicales (CESM)

La Confederación tiene por objeto promover, proteger, difundir y dignificar la afición, enseñanza y práctica de la música, potenciar el asociacionismo civil y apoyar a las Federaciones-miembros en sus tareas y actividades. En colaboración con las Instituciones, ofrecerá a la Sociedad Española actividades culturales y facilitará a toda su población las mejores oportunidades de acceso al bien cultural musical.

Asimismo, asumirá la representación, coordinación y defensa de esfuerzos, aspiraciones e intereses generales de las Federaciones integradas en la Confederación, ante el Gobierno de la Nación o de las diferentes Comunidades Autónomas, en su caso, y demás Instituciones públicas del Estado y de la sociedad civil, y solicitará las ayudas necesarias para continuar la ímproba labor cultural y docente de sus entidades confederadas.

Los fines de esta Confederación tienden a promover el interés general, sus actividades estarán abiertas a los ciudadanos, y no quedarán restringidas exclusivamente a beneficiar a sus asociados [2].

2.2.2 Federaciones de sociedades musicales

Consisten en organismos cuyo objetivo es organizar, ayudar, administrar y albergar a las sociedades musicales a su cargo. Lo más común es que, en España, podamos encontrar una federación de sociedades musicales por comunidad autónoma. Actualmente, según las federaciones obtenidas de la página oficial de la Confederación Española de Sociedades Musicales (CESM), actualmente en España hay 17 federaciones de sociedades, bandas y agrupaciones musicales.

2.2.3 Sociedades (bandas, agrupaciones) musicales

Las Sociedades musicales son entidades privadas, compuestas por socios, que realizan una actividad cultural sin ánimo de lucro, y que han sido creadas con el principal objetivo de fomentar el conocimiento y disfrute de la música. Estas sociedades musicales, aquí en España, deben encontrarse bajo la autoridad de su respectiva Federación musical, dependiendo del lugar del que sea originaria.

Existen sociedades musicales de menor tamaño y recursos, como por ejemplo las bandas. Una banda de música es una agrupación musical formada por los tres tipos de instrumentos: viento, cuerda y percusión. Un factor importante para diferenciar las bandas de las asociaciones musicales es que las bandas deben estar constituidas por instrumentos que puedan ser tocados mientras el músico marcha en los desfiles, en las batallas o en desfiles religiosos.

Las entidades musicales más pequeñas son las agrupaciones musicales, las cuales consisten en dos o más personas que, a través de la voz o de instrumentos musicales, interpretan obras musicales pertenecientes a diferentes géneros y estilos.

2.3 Aplicaciones similares

En este apartado se mencionan algunas aplicaciones que tienen objetivos de mercado similares a Glissandoo, además de proporcionar funcionalidades parecidas e ir enfocadas en parte al mismo tipo de usuario. También se aclarará algunas diferencias entre Glissandoo y cada uno de estos sistemas software de gestión, y se aclarará porqué es necesario un software como Glissandoo, diferente al resto de ellos.

2.3.1 Band.us

Band.us es una aplicación para la comunicación entre líderes de comunidades. Band es una aplicación de gestión de diferentes tipos de comunidades, es decir, no gestiona solamente comunidades o agrupaciones musicales, por lo que tiene funcionalidades genéricas para todo tipo de comunidades. Los ejemplos que muestran su página principal son comunidades para equipos de animación y exploración, y para comunidades de estudiantes y padres. Está disponible como aplicación en Android y iOS.

La principal diferencia entre Glissandoo y Band.us es que Band.us está enfocada a mejorar la comunicación entre líderes de diferentes tipos de comunidades. Es genérica, siempre ofrece las mismas funcionalidades, independientemente de la temática de la comunidad a la que de soporte de comunicación. Por tanto, no le es posible ofrecer funcionalidades específicas de entornos musicales. Funcionalidades que el panel control principal de Glissandoo sí que puede ofrecer, como por ejemplo asignar temas con partituras, vídeos y enlaces pertinentes a orquestas o agrupaciones musicales, y vincular estos temas a eventos o ensayos. Otro ejemplo sería asignar una partitura diferente o la misma a cada instrumento de una agrupación o banda musical.

En el caso de este proyecto, en el cual se ha desarrollado el panel de control de federaciones, no existe mayor diferencia entre las funcionalidades que ofrece y las que ofrece Band.us, más allá de que el panel de federaciones permite cambiar detalles de la federación y consultar las sociedades (y algunos detalles de las mismas) que pertenecen a dicha federación, además de algunas estadísticas de las anteriores.

2.3.2 konzertmeister.app

Esta aplicación web alemana es muy similar a la que proporciona la empresa Glissandoo. De hecho, el panel de control que muestra la página principal es similar al panel de control principal de Glissandoo. Alemania es un objetivo en el enfoque de negocio de Glissandoo, pero la existencia de este software web dificulta la captación de bandas, agrupaciones y sociedades alemanas. Está disponible tanto en *Web aplication* como en aplicación para Android y Apple.

Las principales funcionalidades de konzertmeister son el control de asistencia de músicos a eventos y actos musicales, el chat para comunicar cualquier novedad y, para los líderes o usuarios que vayan a llevar la gestión de estos eventos, ofrece la funcionalidad de crear, editar, detallar información y comunicar eventos, actos y ensayos de todo tipo. Sin embargo, como ya se ha mencionado antes, Glissandoo ofrece la gestión de agrupaciones, bandas, coros, etc. de forma que se puedan crear eventos y actos (funcionalidad que también ofrece konzertmeister), además de asignar temas con partituras, vídeos y enlaces pertinentes a orquestas o agrupaciones musicales, y vincular estos temas a eventos o ensayos. Glissandoo permite gestionar el material musical que se tratará en estos tipos de eventos.

De la misma forma que en el apartado anterior, en lo que tiene que ver con el desarrollo del panel de control de federaciones, konzertmeister no ofrece la funcionalidad de consultar las sociedades o agrupaciones musicales de cualquier federación, sino que se centra en la gestión de eventos y en la comunicación entre los líderes encargados de gestionar estos eventos. Un punto en común entre konzertmeister y el panel de control de federaciones es que ambos pueden mostrar estadísticas. El panel de control de federaciones de Glissandoo puede ofrecer estadísticas extraídas de las sociedades, como el número de músicos, directivos, etc; mientras que konzertmeister puede ofrecer estadísticas de asistencia a los eventos gestionados.

2.3.3 Chorus Connection

Tal como menciona su página web, "esta aplicación es una herramienta en línea que ayuda a los coros a administrar todo en un solo lugar, optimizar sus operaciones, ahorrar tiempo al personal y construir sus comunidades de miembros. Las características típicamente incluyen un directorio de miembros, tablón de anuncios, listas de correo electrónico, almacenamiento de archivos, pagos de miembros, biblioteca de música, calendario, asistencia, gráficos ascendentes y más" [3].

Esta aplicación también es muy similar a Glissandoo, pero está enfocada únicamente a coros de música. La comparación más evidente que podemos extraer entre la aplicación web de Glissandoo y Chorus Connection es la diversidad de colectivos musicales a los cuáles Glissandoo ofrece soporte de almacenamiento de datos, gestión de actos y eventos, etc. Desde federaciones de sociedades musicales, como sociedades musicales, bandas de música, agrupaciones, *big bands,* coros, etc. son los que pueden usar y aprovechar Glissandoo.

Las funcionalidades propiamente dichas son semejantes, pero al ir enfocada a coros, Chorus Connection no ofrece funcionalidades para la gestión de instrumentos dentro del grupo, ni soporte para subir partituras vinculadas a cada instrumento o a todos en general, soporte que el panel de control principal de Glissandoo sí ofrece. El panel de control de federaciones, desarrollado por el alumno, ofrece la consulta de los detalles de la federación del líder que se registre, así como una lista con todas las sociedades musicales pertenecientes a esa federación en cuestión (además de información sobre cada una de las sociedades musicales).

2.3.4 Conclusiones

Como aclaración después de analizar algunas alternativas a Glissandoo y de comparar las funcionalidades proporcionadas por los diferentes paneles de control de Glissandoo (sólo se han mencionado el panel de control principal y el panel de control de federaciones, desarrollado por el alumno) con las funcionalidades que ofrecen el resto de aplicaciones, concluimos que Glissandoo es una aplicación más completa que el resto de las alternativas, pues está desarrollada específicamente para ámbitos musicales y una gran parte de funcionalidades de las alternativas presentadas en conjunto, además de algunas funcionalidades propias de las cuales no hay constancia que otras aplicaciones ofrezcan.

2.4 Tecnologías y herramientas de desarrollo web

En este proyecto se han usado diferentes herramientas de desarrollo. Todas ellas son librerías o *frameworks* externos que se han usado para agilizar y facilitar el desarrollo del trabajo. De la misma forma, se han usado distintas (aunque similares) tecnologías de desarrollo.

La gran mayoría de herramientas y tecnologías que se mencionan en este apartado están enfocadas al desarrollo del *front-end*, es decir, son tecnologías y herramientas de diseño web y de construcción de interfaces. También se verá la herramienta usada para hacer uso del *back-end*, es decir, de servidores, para poder trabajar con datos almacenados en la nube.

2.4.1 Tecnologías de *front-end*

Son aquellas que han permitido desarrollar el código del panel de control y de la aplicación en general. Todas estas han sido usadas usado, y en cada apartado se explica qué lugar ocupa cada tecnología, qué beneficios obtenemos haciéndolas formar parte de la aplicación y cuál es su contexto y papel entre las otras tecnologías. En muchas de estas tecnologías se referencian libros en los que se puede encontrar más información sobre las mismas.

2.4.1.1 Javascript

Javascript es la base del proyecto. En realidad, Javascript es el lenguaje de desarrollo web más difundido, extendido y normalizado en el mundo de la programación web.

Suele definirse como un lenguaje de secuencias de comandos orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico [4].

2.4.1.2 HTML

HTML (*Hyper Text Markup Language*) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Permite estructurar todos los componentes, párrafos, bloques o cualquier otro elemento web en la página de manera muy intuitiva [5].

2.4.1.3 CSS

CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML. CSS describe cómo debe ser renderizado (mostrado) el elemento estructurado en la página web o en cualquier otro contexto.

Estas tres tecnologías son las más importantes de una página web, y entre ellas forman la base de la gran mayoría de las páginas web del mundo [6]. Ahora bien, en este proyecto se han usado otras dos tecnologías más, ambas complementando a Javascript.

2.4.1.4 TypeScript

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. TypeScript puede ser usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor (Node.js).

TypeScript extiende la sintaxis de JavaScript, por tanto, cualquier código JavaScript existente debería funcionar sin problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original [7].

2.4.1.5 React

React es una librería de código abierto de Javascript, y ha sido desarrollada por Facebook. Está diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una o más páginas [8]. En el proyecto se han usado algunos componentes, funciones y *hooks* de React, como por ejemplo *React-intl* (traducción) o *useState* (estados de la aplicación).

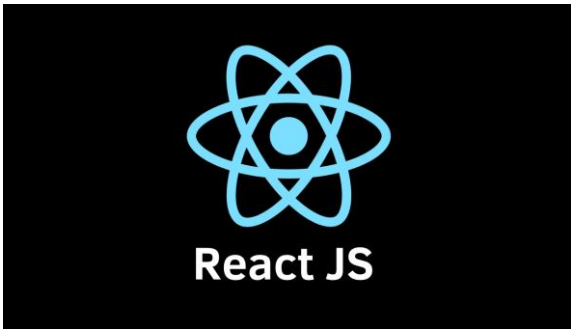


Figura 2 - React JS

2.4.1.6 NextJS

Por último, NextJS es una librería de React desarrollado por Vercel y diseñado para TypeScript. NextJS ofrece a aplicaciones web que usen React funciones para

representar el lado del servidor y la generación de sitios web estáticos [9]. El proyecto del panel de control de federaciones se inicializó como un proyecto de NextJS, con el comando:

```
npx create-next-app next-federations --use-npm
```

Esta librería nos ofrece una de las funciones más importantes del proyecto, *getServerSideProps* (declarada en cada página de la aplicación) para obtener datos desde el servidor y llevarlos al lado del cliente. De esta forma, la carga de los datos externos a la aplicación se realiza en el servidor, y no se ejecutan en el lado del cliente, lo que agiliza en gran medida la interacción y rendimiento de la aplicación.

Figura 3 - NEXT JS

2.4.2 Alternativas a las tecnologías de *front-end* escogidas

Existen diferentes alternativas al desarrollo de *front-end* web mencionadas en el apartado anterior. En este caso, la empresa Glissandoo decidió escoger estas tecnologías para llevar a cabo el proyecto del panel de control de federaciones porque el resto de aplicación web, paneles de control, blog, etc. también se habían desarrollado con estas tecnologías.

El startup decidió que sería una buena práctica mantener una homogeneidad en los diferentes componentes que maneja la web. Sin embargo, por investigación propia, el autor del proyecto ha visto oportuno mencionar algunas tecnologías que pudieron haber sido la alternativa a las usadas finalmente.

2.4.2.1 Angular2

Angular2 es un *framework* para aplicaciones web de TypeScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. En este sentido, es muy similar a React. Incorpora estructuración por componentes.

Algunas de sus principales ventajas son su documentación, que es muy extensa y facilita mucho la consulta en caso de problemas. También permite gran cantidad de lenguajes, como TypeScript, para la estructura de código.

Sin embargo, también cuenta con sus desventajas. Una de ellas es el posicionamiento SEO, que puede llegar a ser un factor tedioso que mejorar. Otro inconveniente es que, a pesar de que soporta otros lenguajes, generalmente se aconseja aprender TypeScript.

En este contexto, Angular2 sería una posible alternativa a la librería React.

Figura 4 - Angular2

2.4.2.1 Vue.js

Vue.js es un *framework* de Javascript relativamente nuevo. Combina dentro de sus características conceptos tanto de Angular2 como de React para construir un framework realmente limpio.

Es capaz de mantener en sincronía los datos del modelo y la vista, este concepto es llamado *data driven view*.

Se trata de un framework sumamente ligero con buen rendimiento y aprendizaje muy rápido. Sin embargo, la documentación fuera de los comienzos es más escasa que las alternativas. Además, es necesario utilizar *plugins* externos como *routing* o manejo de estados, y hay que añadir un *bundler* Webpack o similar [11].

Figura 5 - Vue.js

2.4.3 Herramientas de *front-end*

Para mejorar los estilos, la visibilidad y la interacción del *front-end* del panel de control descrito en este proyecto, se han usado las librerías descritas en este apartado.

2.4.3.1 TailwindCSS

TailwindCSS es un *framework* de CSS que permite un desarrollo ágil que se puede aplicar con facilidad en el código HTML y unos flujos de desarrollo que permiten optimizar mucho el peso y la organización del código CSS.

Se decidió usar este framework para liberar al proyecto de la gran cantidad de archivos, carpetas y, en general, estructura que el código CSS requiere. El método común de uso de CSS es mediante archivos específicos de extensión CSS, los cuales contienen código CSS. Cada archivo que quiera hacer uso de esos estilos expresados mediante código, tendrá que ser importados en cada componente, página o archivo en general en el que quieran ser usados. Esto suele resultar en que cada componente suele tener su archivo CSS, de forma que cuantos más componentes y páginas vayan surgiendo en el proyecto, más grande será la carga de archivos CSS.

TailwindCSS evita esta sobrecarga de estructura en el proyecto, eliminando la necesidad de crear archivos CSS para aplicar estilos a los componentes y al código HTML. Instalando y configurando adecuadamente TailwindCSS en cualquier proyecto, mediante los atributos *className* de los elementos HTML se pueden proporcionar estilos a los mismos. Con ayuda de la documentación de TailwindCSS, podemos usar palabras clave para referirnos a propiedades de estilo CSS. Ejemplo de uso de TailwindCSS:

```
<a
    className="text-2xl text-white mx-3 inline-block"
    href="https://twitter.com/glissandoocom"
>
    <TwitterOutlined />
</a>
```

Figura 6 - Ejemplo de uso de TailwindCSS

Figura 7 - TailwindCSS

2.4.3.1 Ant Design

Ant Design es una interfaz de usuario diseñada para aplicaciones web. Se trata de un conjunto de componentes React de alta calidad listos para uso rápido. Ha sido desarrollado en TypeScript con tipos estáticos. Ant Design ofrece un paquete de recursos de diseño y herramientas de desarrollo muy útiles para desarrolladores. Un detalle muy importante y que ha sido realmente útil en este proyecto es la potente personalización del tema (*theme*) de Ant, así como de sus diferentes componentes.

Entre los componentes de Ant que se han usado en este proyecto se encuentran el Ant Form (para los formularios), Layout (para la estructura principal del panel de control) y el Upload (para subir archivos de cualquier tipo).

Figura 8 - Ant Design

2.4.3.1 Redux

Redux es un contenedor de estado de aplicaciones JavaScript. Es muy útil para escribir aplicaciones que se comportan de manera consistente, se ejecutan en distintos ámbitos (cliente, servidor y nativo), y son fáciles de probar. Además, provee una experiencia muy cómoda de desarrollo, gracias a la edición en tiempo real.

Es posible usar Redux combinado con React, o cualquier otra librería de vistas. Es una librería muy pequeña y no tiene dependencias.

Figura 9 - Redux

2.4.4 Tecnologías de *back-end*

En el proyecto especificado, se hace necesario el uso de un *back-end*. Por ello, se usan estas librerías mencionadas a continuación que permiten la autenticación y el almacenamiento de datos en tiempo real [12].

2.4.4.1 Autenticación - Firebase

La mayoría de las aplicaciones necesitan conocer la identidad del usuario que las usa. Cuando se conoce la identidad del usuario, la aplicación puede almacenar de forma segura los datos del usuario en la nube y proporcionar la misma experiencia personalizada en todos los dispositivos disponibles para el usuario.

Firebase Authentication proporciona servicios de *back-end*, un SDK fácil de usar y una biblioteca de IU prediseñada para usar tu aplicación a fin de autenticar a los usuarios. Admite la autenticación mediante correo electrónico, contraseña, número de teléfono, proveedores de identidad populares (como Google, Facebook y Twitter).

Figura 10 - Firebase Authentication

2.4.4.2 Cloud Firebase - Firebase

Cloud Firestore es una base de datos flexible y escalable para el desarrollo móvil, web y de servidores de Firebase y Google Cloud. Mantiene los datos sincronizados en las aplicaciones cliente a través de componentes de escucha en tiempo real y ofrece soporte sin conexión para dispositivos móviles y web para crear aplicaciones receptivas que funcionen independientemente de la latencia de la red o la conectividad a Internet.

En este proyecto, Cloud Firestore ha servido de base de datos y almacén de todos los datos tratados en el panel de control. Esos datos incluyen los datos de las federaciones y de las sociedades musicales.

Figura 11 - Cloud Firestore

3. Elicitación de requisitos

3.1 Introducción

3.1.1 Propósito

La sección de elicitación de requisitos tiene como finalidad lo que se demanda y espera del panel de control, algo esencial para el desarrollo del mismo.

3.1.2 Ámbito

En vista de la precariedad y la falta de digitalización y centralización de la gran mayoría de gestiones de las sociedades, agrupaciones y bandas musicales se ha decidido crear una aplicación donde la gestión de eventos, ensayos y tratamiento de la información de las diferentes agrupaciones pueda llevarse a cabo desde una misma herramienta.

En el caso en específico del panel de control de federaciones, ese proyecto resuelve las dificultades que puede conllevar el seguimiento y la monitorización de cada agrupación o sociedad individualmente bajo el dominio de cada federación musical.

3.1.3 Definiciones, acrónimos y abreviaturas

- Músico/a: persona que toca un instrumento musical o compone música, especialmente si esta persona se dedica a ello profesionalmente.
- Federación de sociedades musicales: organismos cuyo objetivo es organizar, ayudar, administrar y albergar a las sociedades musicales a su cargo.
- Sociedades musicales: entidades privadas, compuestas por socios, que realizan una actividad musical, y que han sido creadas con el principal objetivo de fomentar el conocimiento y disfrute de la música.
- Agrupaciones musicales: dos o más personas que, a través de la voz o de instrumentos musicales, interpretan obras musicales pertenecientes a diferentes géneros y estilos.
- Banda musical: agrupación musical formada por los tres tipos de instrumentos. Las bandas musicales están constituidas por instrumentos que pueden ser tocados mientras el músico marcha en cualquier tipo de evento.

- Instrumento musical: objeto formado por una o varias piezas que se usa para producir música.
- Evento (cultural): son aquellos que involucran como temática principal alguna rama del arte o tradiciones. En el caso de este proyecto, son eventos musicales.
- Ensayo musical: es el espacio en el cual los músicos tienen la oportunidad de conocer en diferentes formas su propio repertorio, desmenuzarlo, analizarlo, interiorizarlo y perfeccionarlo.
- Líder (de agrupación musical): persona encargada de dirigir, gestionar y tomar las decisiones oportunas de una agrupación musical.
- Socio: persona que pertenece a una agrupación o sociedad musical.
- Asistencia: acción de asistir a un ensayo o a un acto.
- Repertorio: conjunto de obras musicales que una banda o una agrupación tiene preparadas para representar.
- Perfil: entorno personalizado para un individuo o un grupo de personas que se desarrolla de acuerdo a sus preferencias de configuración.
- Usuario: son los clientes que usan habitualmente ciertos programas, aplicaciones y sistemas de un dispositivo informático.

3.1.4 Visión global

El propósito en específico del panel de control a desarrollar por el alumno es gestionar y monitorizar las actividades de las sociedades musicales. Entre otras funcionalidades, las cuales están restringidas a líderes de federaciones de sociedades musicales, se encuentran acceder a la lista de sociedades musicales bajo el dominio de la federación a la cual pertenece el líder, la posibilidad de acceder a cada sociedad y de obtener algunos datos de las mismas, editar la información de la federación a la cual pertenece el usuario (logo de la federación, nombre, teléfono, correo electrónico, entre otros), consultar la lista de eventos de la federación y gráficas respecto a los mismos y generar enlaces que permitan la conexión y cesión de datos de las sociedades hacia la federación.

Con respecto a la aplicación web en general, se compone de una web pública (replicada por el alumno en la primera parte de las prácticas en el startup) desarrollada con las mismas tecnologías mencionadas en este proyecto. En la parte privada de la

aplicación web, se encuentra el panel de control principal, el cual permite las principales y más importantes funcionalidades de gestión de actos, ensayos, repertorios, miembros y todo lo relacionado con las agrupaciones y bandas musicales.

Por último, recientemente se ha desarrollado, implementado e integrado al panel de control principal (*dashboard*) una sección de socios, en el cual se podrán crear planes de pago para sociedades, las cuales serán parte también del panel de control de federaciones.

3.2 Descripción general

3.2.1 Perspectiva del producto

La aplicación web de Glissandoo, salvo la parte pública (en la que los usuarios de internet pueden consultar información sobre la aplicación, Glissandoo o una demo de la aplicación), en su gran mayoría es privada. Esto limita el acceso a las diferentes secciones del panel de control a aquellos usuarios que sean clientes de Glissandoo, es decir, líderes o directivos de las agrupaciones musicales registradas. Un factor importante es que la aplicación web está disponible en cualquier tipo de navegador en cualquier tipo de dispositivo.

Por otra parte, el panel de control de federaciones desarrollado por el alumno es completamente de ámbito privado, a excepción de la página de inicio de sesión.

3.2.2 Características de usuario

Podemos distinguir entre varios tipos de usuario que pueden acceder a la aplicación web:

- Anónimo: persona no identificada que visita la página web www.glissandoo.com.
- Usuario registrado: persona registrada previamente en el sistema que ha introducido satisfactoriamente sus credenciales. Existen diferentes tipos de usuario:
    - Músico / miembro de la agrupación musical: usuario estándar que hace uso de las herramientas de gestión para consultar fechas de eventos o ensayos, para confirmar asistencia, y algunas funcionalidades más.
    - Administrador: usuario con permisos especiales que podrá llevar a cabo las principales tareas de gestión. tales como organizar eventos, ensayos, crear y modificar grupos, gestionar los miembros de los grupos, etc.

- Líderes de federación: usuarios con permisos especiales que, además de los anteriores, poseen permisos para acceder al panel de control de federaciones. en el que pueden consultar información sobre las sociedades bajo el dominio de su federación, así como editar los datos de la federación a su cargo.

3.2.3 Operaciones permitidas

En este apartado se enumeran las diferentes operaciones permitidas y las distintas funcionalidades disponibles para cada uno de los usuarios mencionados en el apartado anterior.

Anónimo: El usuario no registrado puede visitar la página web pública www.glissandoo.com. Además, podrá solicitar una demo del producto, así como ponerse en contacto con el CEO y CTO del startup mediante un formulario

Músico / miembro de la agrupación musical: Puede hacer uso de las herramientas de gestión para consultar fechas de eventos o ensayos, para confirmar asistencia, etc.

Administrador: Pueden organizar (crear, editar, eliminar, notificar a los músicos) eventos, ensayos, grupos, gestionar los miembros de los grupos, crear temas mediante enlaces o archivos de audio, añadir esos temas al repertorio, crear comunicaciones y vincularlas a usuarios en específico o a grupos, etc.

Líderes de federación: Pueden consultar una lista de las sociedades bajo el dominio de su federación, consultar información y datos sobre cada una de las sociedades (en función de si estas han consentido la cesión de los datos a Glissandoo), así como editar los datos de la federación a su cargo, consultar los eventos de la federación y detalles de los mismos o generar enlaces para que las sociedades puedan conectarse a Glissandoo y ceder sus datos para su futura consulta.

3.2.4 Restricciones generales

Para hacer uso de la aplicación web, así como los diferentes paneles de control de la parte privada, con sus funcionalidades al completo y una adecuada presentación se recomiendan los navegadores web de Google Chrome, Mozilla Firefox, Safari o Microsoft Edge, aunque es posible navegar por la aplicación mediante Internet Explorer asumiendo un menor rendimiento. El cliente implementado en React genera archivos JavaScript, de los cuales el propio navegador se encargará de renderizar y ejecutar las operaciones que sean pertinentes sin necesidad de ninguna herramienta o complemento adicional en el navegador.

3.2.5 Supuestos y dependencias

Se utiliza Firebase de Google como *back-end* que dispone de servicios usados en este proyecto como la facilitación de autenticación que proporciona la propia Firebase Authentication, el alojamiento de datos en la Cloud Firestore y el uso de Firebase Functions, con las cuales se han podido implementar *endpoints* (funciones implementadas en el servidor para ser usadas en el front-end) que han reducido la carga de código en el proyecto.

Existe un plan gratuito para el servicio de Firebase, pero, por desgracia, no se adecua correctamente a este proyecto. El uso de la Firebase Firestore con una gran cantidad de datos almacenados y, en especial, el uso de Firebase Functions, han hecho necesario el plan de pago de esta herramienta.

3.3 Requisitos específicos

3.3.1 Requisitos de interfaces externas (no funcionales)

3.3.1.1 Interfaces de usuario

La aplicación web se mostrará, como de costumbre, en la pantalla del usuario, de forma intuitiva y sencilla

3.3.1.2 Interfaces hardware

En la aplicación web de Glissandoo, así como el panel de control de federaciones, no se ha usado ningún hardware en específico que sea requerido para el uso de las mismas.

3.3.1.3 Interfaces software

La aplicación está desarrollada mediante la librería de código de JavaScript: React, usando como base la tecnología Next.JS, usando componentes de React y TypeScript. Puede ser visitada desde cualquier dispositivo, a pesar de que se recomienda el navegador web de escritorio para acceder a los paneles de control.

La base de datos se ha montado en Firebase, un *back-end* que sirve y almacena una base de datos con estructura json. Se ha establecido un modelo de datos desde la aplicación en *front-end*.

3.3.2 Requisitos funcionales

Los requisitos funcionales son una descripción del servicio que debe ofrecer el software. Normalmente, describe un sistema software o cualquiera de sus componentes como un conjunto de entradas al sistema software, su comportamiento y las salidas que presenta. Pueden ser cálculos, tratamiento de datos, alguna interacción con el usuario o cualquier otra funcionalidad específica que defina qué función debe realizar un sistema.

En este caso, se mencionan únicamente los requisitos funcionales encontrados y analizados para el desarrollo del panel de control de federaciones.

ID	Nombre del requisito	Descripción	Prioridad (1-5)
[RF1]	Iniciar sesión	El líder de la federación podrá iniciar sesión en el panel de control de federaciones con un usuario y una contraseña. Se mostrará una notificación de error si los datos introducidos no son válidos.	5

[RF2]	Redirección a la pantalla principal	Una vez autenticado el usuario se redirigirá a la pantalla principal del panel de control de federaciones con los datos del usuario cargados.	5
[RF3]	Consultar lista de sociedades musicales	Lo primero que el usuario autenticado verá cuando acceda al panel de control será la lista (en formato de tabla) de sociedades musicales que se encuentran bajo el dominio de su federación.	5
[RF4]	Mostrar detalles de las sociedades musicales en la tabla de sociedades	Entre los detalles que se podrán observar, además del nombre de la federación, es la localidad en la que se encuentra, el número de directivos, músicos y socios que tiene, su logo y un *check* verde en caso de que la sociedad haya conectado los datos a la federación y Glissandoo. Requisitos relacionados: **[RF3]**	4
[RF5]	Consultar estadísticas de la federación	Se mostrarán algunas estadísticas como el número de sociedades musicales que tiene la federación en total, el número de músicos que podemos encontrar en la federación y el número de socios que tiene la federación en total.	3
[RF6]	Filtrar búsqueda de resultados en la tabla de sociedades	Se podrá buscar por nombre de sociedad, apareciendo así las sociedades que se correspondan con la búsqueda realizada. Requisitos relacionados: **[RF3]**	4
[RF7]	Ordenar los valores de la tabla en orden	Se podrá ordenar la tabla de sociedades ascendente y descendentemente según los valores numéricos que contenga la	2

	ascendente y descendente	tabla, esto es, los directivos, los músicos y los socios.	
		Requisitos relacionados: **[RF3]**	
[RF8]	Cerrar sesión	El usuario podrá cerrar su sesión, y no se podrá acceder de nuevo a los datos hasta que se vuelva a iniciar sesión.	5
[RF9]	Redirección a la página de autenticación	Si el usuario autenticado cierra su sesión, la página redirigirá a la página de inicio de sesión. En caso de que un usuario no autenticado tratara de acceder a una página del panel de control, con independencia de la página a la cual se quiera acceder, el servidor redirigirá a la página de autenticación.	5
		Requisitos relacionados: **[RF8]**	
[RF10]	Editar información de la federación	En la sección Ajustes, el usuario podrá editar el nombre, correo electrónico, teléfono, web y dirección de la federación en campos de texto.	5
[RF11]	Cambiar foto de la federación	Se podrá subir una foto al panel de control para cambiar la foto actual de la federación.	4
		Requisitos relacionados: **[RF10]**	
[RF12]	Acceder a los detalles de las sociedades de la tabla	Acceder a una página donde se mostrarán todos los datos almacenados de la sociedad seleccionada.	5
[RF13]		Igual que en la página principal del panel de control, se mostrarán algunas	3

	Consultar estadísticas de la sociedad	estadísticas, pero esta vez pertenecientes a la sociedad. Las estadísticas contarán con el número de directivos y músicos que podemos encontrar en la sociedad, con su respectiva gráfica de diversidad de género, y el número de socios que tiene la sociedad. Requisitos relacionados: **[RF12]**	
[RF14]	Ocultar algunos datos de la sociedad (en caso de que no esté conectada con la federación)	Si la sociedad seleccionada no se ha conectado ni con la federación ni con Glissandoo, se mostrarán los datos más importantes de la misma, pero no aparecerán datos como las estadísticas o eventos. Requisitos relacionados: **[RF12]**	4
[RF15]	Generar enlace para conectar los datos	Se podrá generar un enlace para enviar a cualquier sociedad musical (funcionalidad de enviar no incluida por el momento) para que la sociedad pueda conectarse a Glissandoo con su federación. Requisitos relacionados: **[RF12]**	5
[RF16]	Conectar sociedad musical con la federación y con Glissandoo	En una página pública, a la cual se accede con el enlace antes mencionado, se informará de los datos que se compartirán en caso de conectar. La página cuenta con un botón que conectará la sociedad con su federación y con Glissandoo.	4
[RF17]	Acceder a la sección de Eventos	Se podrá acceder a otra de las tres páginas principales mediante la barra lateral. Se accederá a Eventos, donde se	5

		puede encontrar información relacionada con los conciertos programados en la sociedad	
[RF18]	Mostrar gráfica de eventos del último año	El usuario podrá consultar una gráfica en la que se verá la evolución del número de conciertos que se celebran cada mes durante los últimos doce meses. Requisitos relacionados: **[RF17]**	4
[RF19]	Consultar lista de conciertos próximos y anteriores	El usuario podrá tener acceso a una lista de eventos. Podrá cambiar el tipo de vista de Anteriores (ya ocurridos) a Próximos. La lista mostrará los detalles del evento y la sociedad a la que pertenece cada uno Requisitos relacionados: **[RF17]**	5
[RF20]	Mostrar calendario con eventos programados	Se podrá interactuar con un calendario que mostrará un punto azul debajo de cada día que tenga programado un evento. Requisitos relacionados: **[RF17]**	4

3.3.3 Restricciones del diseño

Algunas partes del proyecto estarán construidas y optimizadas con componentes de comportamiento *responsive.* Sin embargo, esto no será así en todas las páginas del panel de control. Por esa razón, se recomienda el uso de dispositivo de escritorio (PC, ordenador portátil, etc.) para su uso.

3.3.4 Atributos

3.3.4.1 Seguridad

El panel de control se ha diseñado para que, una vez se intente realizar acceso a cualquier parte del mismo, el servidor previamente consultará los datos de Firebase para comprobar que el usuario esté registrado. Si el usuario no está registrado, o no se detectan las credenciales necesarias que acrediten que el usuario es un líder de federación, la aplicación redirigirá automáticamente a la página de inicio de sesión. De esta forma, se impedirá el paso a cualquier usuario no registrado o que no tenga los permisos necesarios.

La aplicación tampoco mostrará públicamente los datos personales registrados en los perfiles de los usuarios, por el bien y la protección de estos datos personales. En el propio panel de control, el usuario líder de la federación solamente podrá editar datos de su federación, por lo que no podrá modificar datos de las sociedades musicales.

3.3.4.2 Mantenimiento

El *back-end*, gracias al uso de Firebase, no requerirá de mantenimiento adicional para mantener la funcionalidad, sin importar cuánto tráfico de datos pueda llevarse a cabo en las consultas y uso del panel de control, ya que, al contar con a la versión contratada de Firebase, no existe un límite que se deba vigilar.

Por otra parte, el *front-end* podrá ser modificado, actualizado y periódicamente optimizado con independencia del *back-end*, modificando únicamente los componentes de las librerías React y NextJS. Cualquier modificación o actualización en la parte del cliente de la aplicación (cambios de diseño, cambios o añadidos de funcionalidades o cambios en la seguridad) deberán de llevarse a cabo en la versión de desarrollo (*dev*), volver a compilarlo en modo de producción y subirlo de nuevo al dominio de la página de la empresa, reemplazando la versión anterior.

4. Análisis del sistema

El siguiente apartado detalla la fase de análisis del sistema. En este capítulo se analiza la aplicación, es decir, el panel de control, que se va a desarrollar y se describe la estructura y funcionalidad del mismo mediante diagramas UML, haciendo más fácil la comprensión de lo implementado y cada una de sus funciones.

Al final del capítulo tendremos un modelo con los posibles actores que interactúan con los objetos y componentes del sistema.

4.1 UML

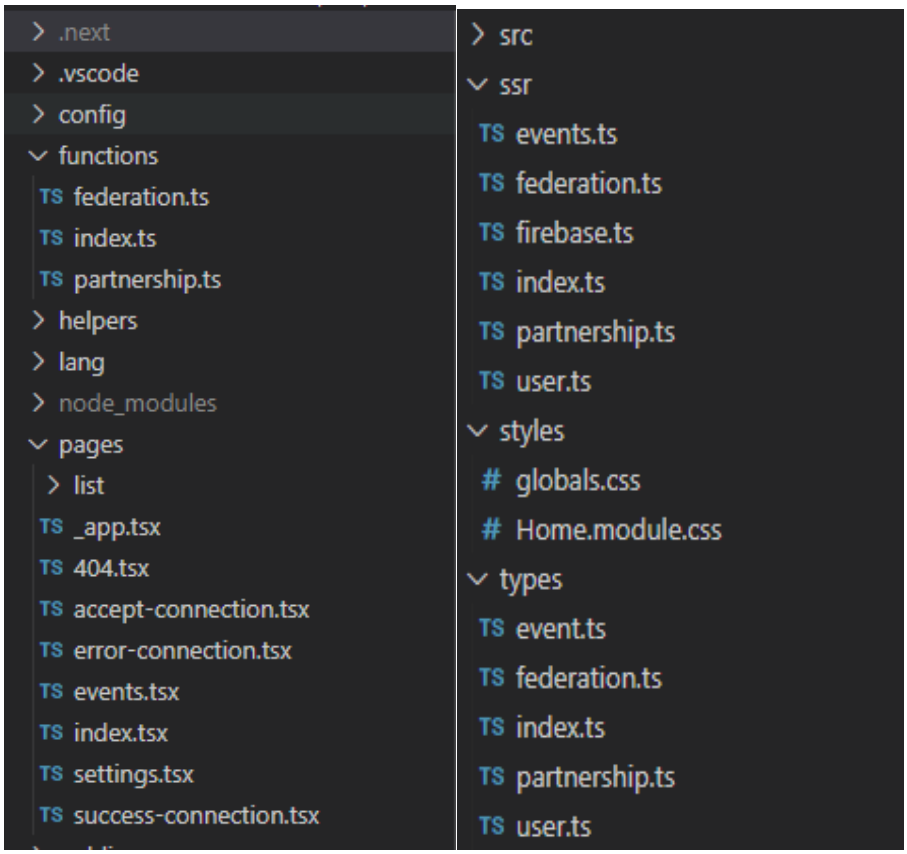
UML (Unified Modeling Language) es un lenguaje de modelado que en estos apartados usaremos para describir nuestro sistema, para detallar e implementar las funciones y componentes que estarán presentes en él. Realizaremos varios diagramas para entender mejor nuestro panel de control.

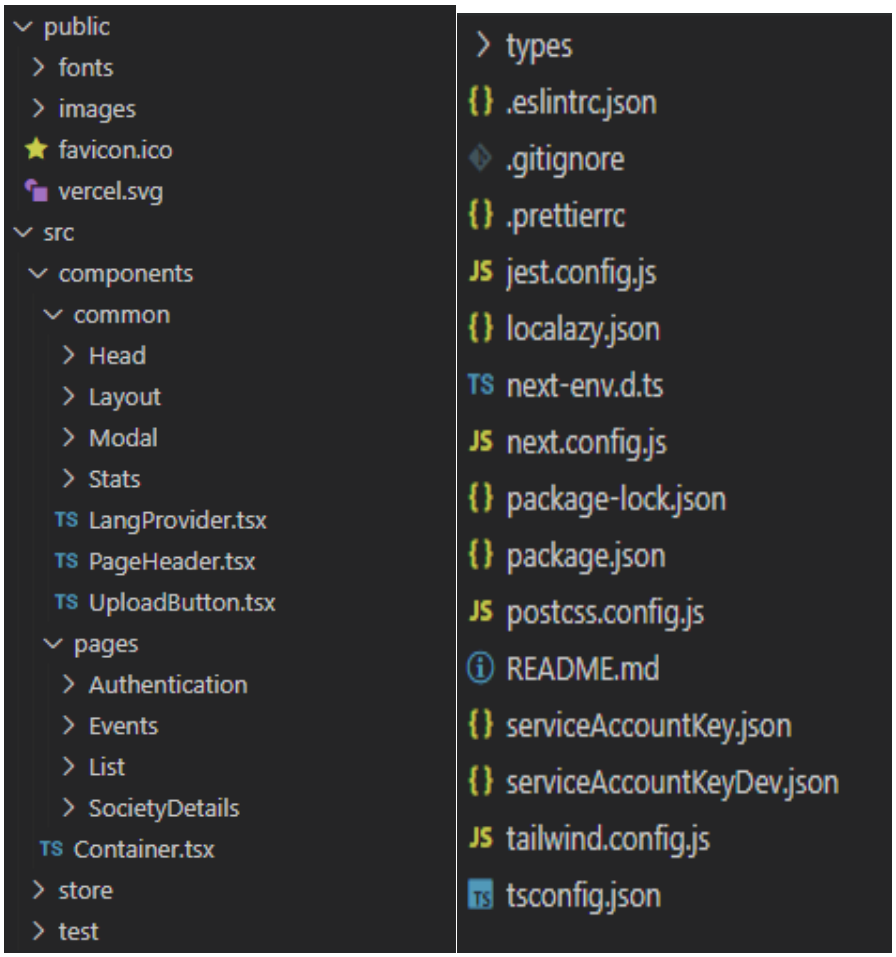
4.2 Diagrama de clases

Un diagrama de clases describe la estructura del sistema al que se hace referencia, mostrando sus clases, funciones y componentes, y las relaciones entre ellos. También consiste en mostrar un diseño conceptual de la información que se maneja y trata en el sistema.

En este proyecto, la aplicación *front-end* se desarrolla en ReactJS y su filosofía, igual al estilo usado en Glissandoo, dicta enfocar como componentes los elementos de la aplicación. Dichos componentes se agrupan en módulos, los cuales siguen una jerarquía.

En las siguientes imágenes se muestra visualmente la estructura y jerarquía del proyecto. Las dos imágenes de la columna de la izquierda representan componentes funcionales y páginas, y las dos imágenes que forman la columna de la derecha representan los archivos de utilidad esporádica y configuración del proyecto.


.next
.vscode
config
functions
federation.ts
index.ts
partnership.ts
helpers
lang
node_modules
pages
list
_app.tsx
404.tsx
accept-connection.tsx
error-connection.tsx
events.tsx
index.tsx
settings.tsx
success-connection.tsx
src
ssr
events.ts
federation.ts
firebase.ts
index.ts
partnership.ts
user.ts
styles
globals.css
Home.module.css
types
event.ts
federation.ts
index.ts
partnership.ts
user.ts


public
fonts
images
favicon.ico
vercel.svg
src
components
common
Head
Layout
Modal
Stats
LangProvider.tsx
PageHeader.tsx
UploadButton.tsx
pages
Authentication
Events
List
SocietyDetails
Container.tsx
store
test
types
.eslintrc.json
.gitignore
.prettierrc
jest.config.js
localazy.json
next-env.d.ts
next.config.js
package-lock.json
package.json
postcss.config.js
README.md
serviceAccountKey.json
serviceAccountKeyDev.json
tailwind.config.js
tsconfig.json

A continuación, se puede observar un diagrama relacionando los elementos conceptuales de la aplicación.

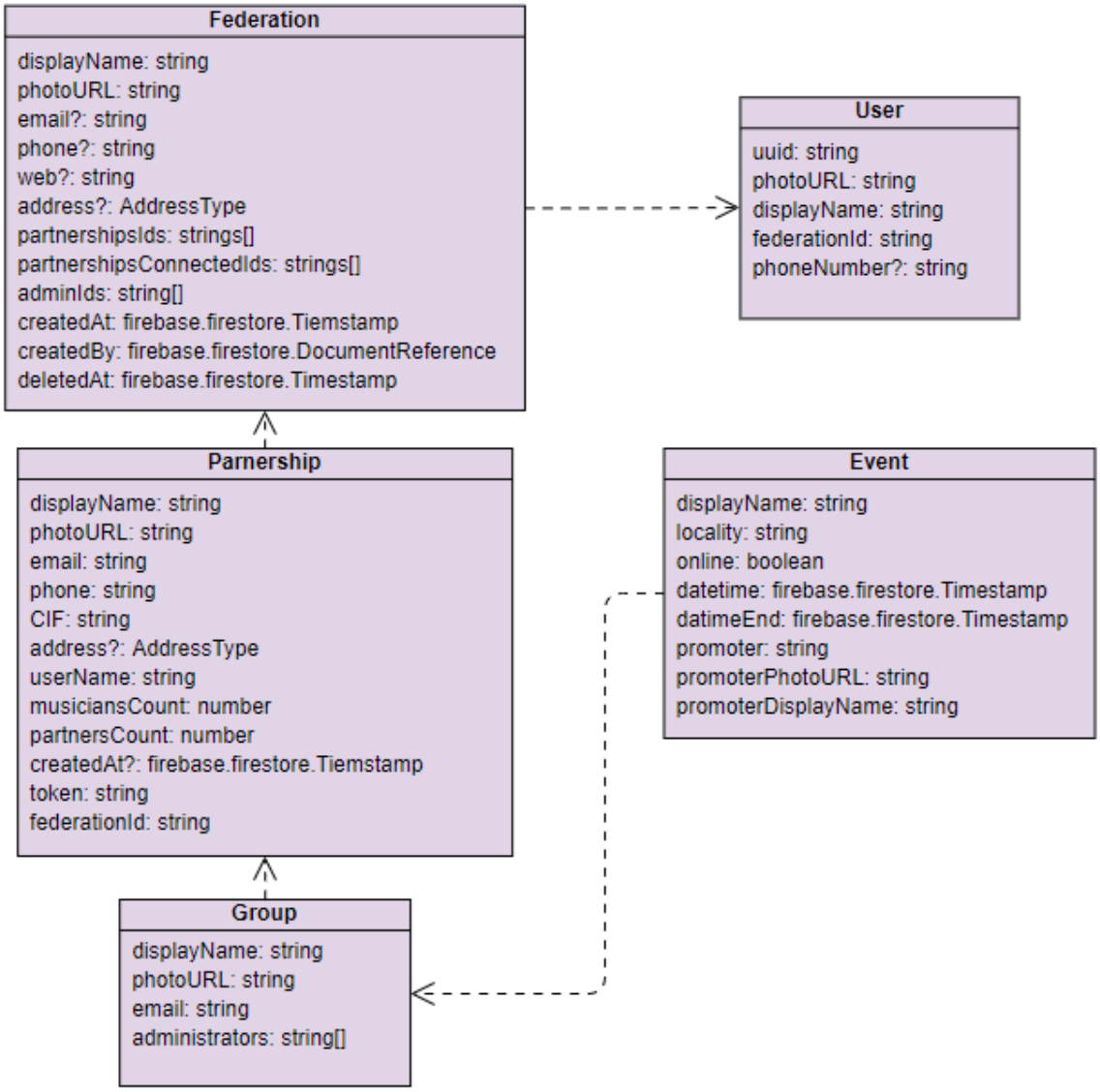


Figura 12 - Diagrama de clases

4.3 Diagramas de casos de uso

Los diagramas de casos de uso describen el comportamiento del panel de control y ponen énfasis en lo que el sistema modelado debe realizar y en qué circunstancias. En este apartado solamente se muestran dos diagramas, ya que solo existen dos posibles actores en la interacción con la aplicación desarrollada y detallada hasta ahora, y se describe qué podrán hacer usando el sistema. Los actores son los siguientes:

ACT-01	**Líder o administrador de la federación**
Autor	Bruno Díaz
Descripción	El líder o administrador de la federación es el único tipo de actor que puede acceder al panel de control.
Comentarios	Es el único que es capaz de usar las funcionalidades del panel de control.

ACT-02	**Líder o administrador de la sociedad musical**
Autor	Ricardo Castillo
Descripción	El líder o administrador de la sociedad musical es el usuario que tiene permisos de administrador de alguna sociedad en el panel de control principal de Glissandoo
Comentarios	Este actor recibe un correo con un enlace a una página desarrollada en este proyecto. En esta página, podrá conectar los datos de su sociedad con la federación a la que pertenezca la sociedad.

4.3.1 Caso de uso para líderes de federación

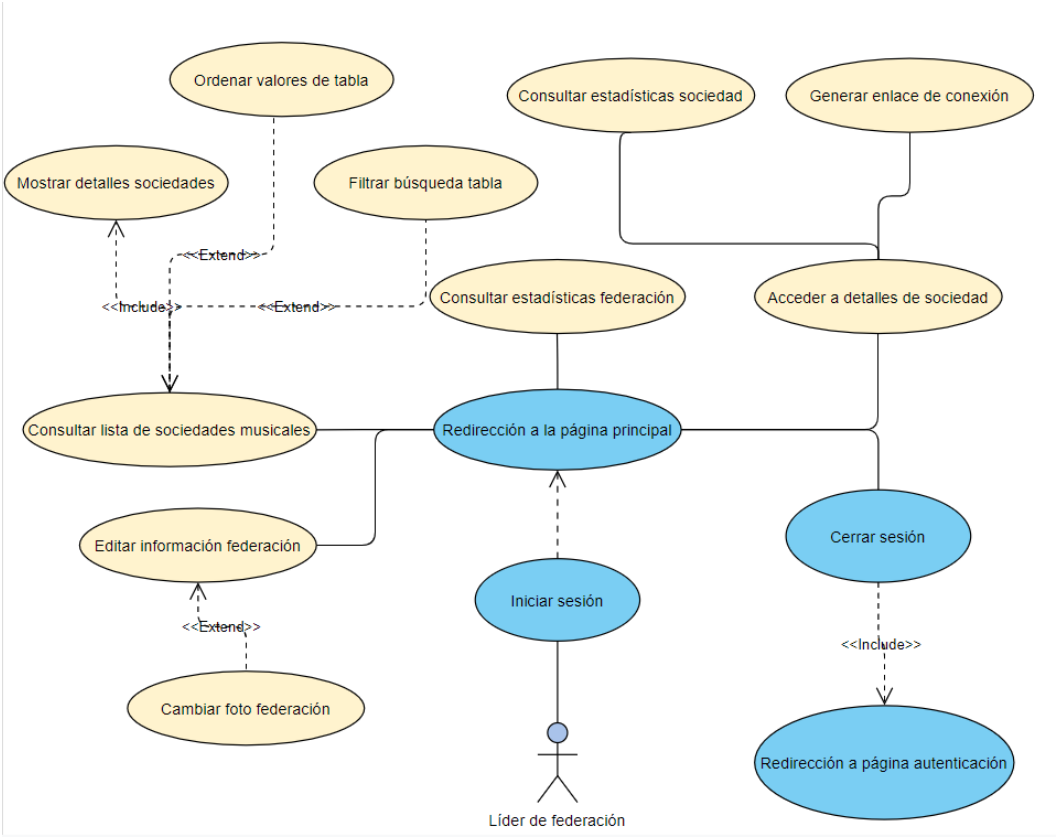


Figura 13 - Diagrama de casos de uso (líder de federación)

4.3.2 Caso de uso para líderes o admins de sociedades

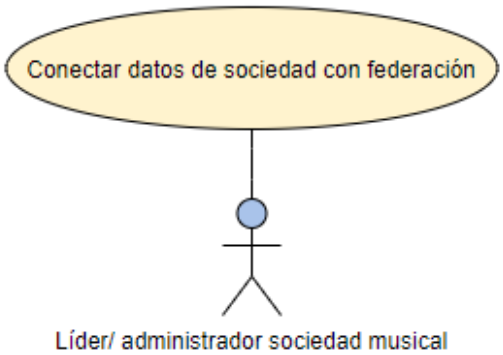


Figura 14 - Diagrama de casos de uso (líder de sociedad)

5. Diseño e implementación

En esta sección se hablará de los elementos usados para diseñar la estructura del proyecto, así como recordar y mencionar detalles relevantes de las tecnologías usadas y explicar, visual y funcionalmente, en qué consiste el panel de control y su uso. También se explicará la separación entre *back-end* y *front-end* y cómo se ha construido cada uno. Cabe mencionar que la completa implementación del panel de control, aunque formara parte de un equipo de desarrollo, ha sido realizada únicamente por el alumno. Se puede destacar el trabajo de apoyo, guía y consulta del CTO de Glissandoo, que ha ayudado al alumno a resolver algunas dificultades, pero la íntegra implementación del panel es obra del alumno.

5.1 Diseño de la arquitectura del sistema

El panel de control de federaciones está basado en el modelo Cliente/Servidor. Los usuarios realizan peticiones al servidor web, que le ofrece una respuesta. Mediante esta arquitectura los accesos, recursos e integridad de los datos usados en el panel de control son controlados por el servidor, de manera que los usuarios pueden acceder únicamente a aquellos datos que estén autorizados a acceder.

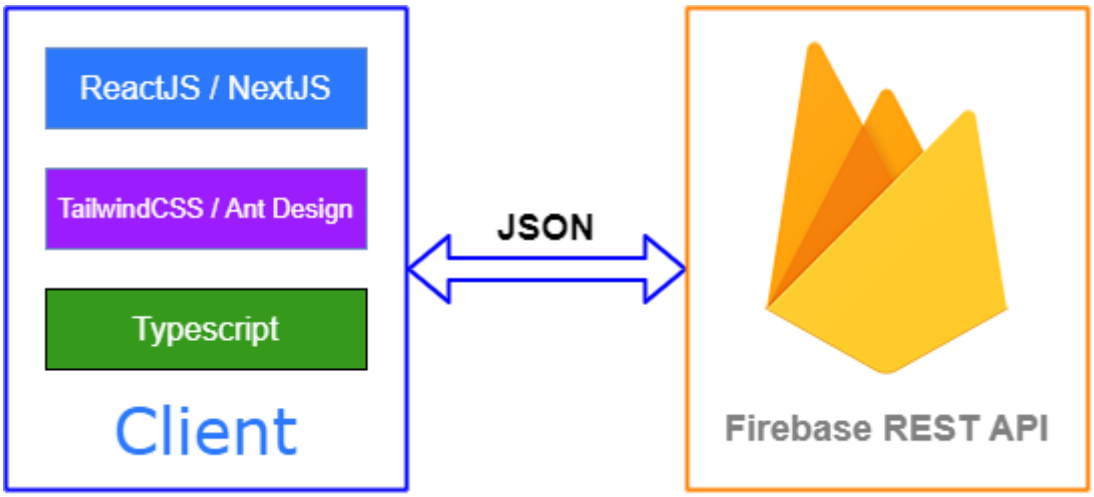


Figura 15 - Diseño de arquitectura

5.1.1 Niveles de arquitectura

- **Nivel de cliente:** esta capa de la arquitectura es la encargada de la renderización de las vistas que se visualizarán en el panel de control, así como la lógica que transformará los datos que se piden a la API REST de Firebase. La capa de cliente la construye la aplicación en NextJS y ReactJS.

- **Nivel de servicio:** En este proyecto se usa Firebase, un *BaaS* (*Back-end as a Service*). Realiza las funciones de API que recibe peticiones de datos (como los datos de la federación o de las sociedades) y devuelve en archivo JSON con la información solicitada. Además, es la piedra angular de la autenticación en este proyecto. En esta capa encontramos, también, la capa de negocio, la base de datos no relacional (Firestore) con estructura de árbol JSON, y es el principal medio de almacenamiento de datos.

En el capítulo 2, Estado del arte, se justifica la elección de estas tecnologías frente a otras alternativas.

5.1.2 Diseño del *front-end* en ReactJS y NextJS

5.1.2.1 Páginas

Una de los tipos de archivo y “fragmentos de código” más útiles en React son las páginas. Bajo el nombre de la carpeta *pages* encontramos estos archivos, los cuales contendrán toda la información que se mostrará por pantalla al visitar la página en el navegador. El nombre de la página determinará la dirección URL desde la raíz mediante la cual se accede al contenido de la misma. Este tipo de archivos son más importantes que los componentes ordinarios usados a lo largo del proyecto.

Debido a que el proyecto tiene como base de implementación NextJS, estas páginas cuentan con una función increíblemente útil: *getServerSideProps()*, que recibe como argumento el contexto actual de la aplicación, ofrecido por NextJS. Esta función

nos permite implementar el SSR (*Server Side Rendering*) en nuestro proyecto. Todas las operaciones, invocaciones o llamadas a APIs que se realicen en la implementación de esta función se realizarán del lado del servidor, evitando cargas problemáticas del lado del cliente al cargar la página. La función *getServerSideProps()* puede retornar valores (*props*) que podemos obtener de las operaciones o llamadas anteriormente mencionadas, y recibirlos como argumentos en la definición de la página, de forma que podamos contar en el lado del cliente con esos valores obtenidos en el lado del servidor.

5.1.2.2 Componentes

Los componentes, también definidos como *Function Components* (Componentes funcionales) dentro del proyecto, son paquetes de código independiente y reutilizables. Tienen el mismo propósito que las funciones de JavaScript, pero funcionan de forma aislada y devuelven HTML a través de una función *render*.

Esta función renderiza las etiquetas HTML necesarias para estructurar el proyecto y permitir su correcto funcionamiento. Gran parte de estas etiquetas serán directamente declaradas como HTML, mientras que algunas etiquetas serán otros componentes funcionales definidos en la aplicación. En definitiva, el principal objetivo del elemento componente es controlar el flujo de datos que se mostrará en la página.

5.1.2.3 Servicios

Los servicios es el lugar donde se debe colocar el código únicamente TypeScript (es decir, sin mezclar con HTML) que se usa para ejecutar cierta lógica de la aplicación. Gran parte de estas funciones se encuentran en el almacenamiento de Firebase (*Functions*), desde donde se llaman y se usan. Sin embargo, algunos otros servicios se encuentran ubicados en el propio proyecto. Estos servicios consisten en algunas llamadas que se hacen desde el lado del servidor.

5.1.3 Diseño de pantallas

5.1.3.1 Componentes Layout de Ant Design

A lo largo del proyecto, se ha hecho un gran uso de Ant Design y los componentes que ofrece. Dependiendo de las necesidades, el equipo técnico de Glissandoo decidió que el uso de componentes ya implementados agiliza en gran medida la implementación

del panel de control. En este apartado se mencionan varios de esos componentes, que han sido claves en la realización e implementación.

- *Table:* este componente es usado en la página principal del panel de control de federaciones. Este componente permite distribuir en columnas (Column, subcomponente de Table) las diferentes propiedades y datos que tendrán las sociedades representadas en esta tabla. Estos subcomponentes columna permiten ordenar los datos en orden ascendente o descendente numéricamente o alfabéticamente, dependiendo del dato contenido en la columna. La columna del nombre de la sociedad contiene un panel de búsqueda para filtrar por nombre.

Figura 16 - Componente *Table*

- *Calendar*: este componente es usado en la sección Eventos para representar con un punto azul que ese día tiene un evento registrado. Es un componente informativo.

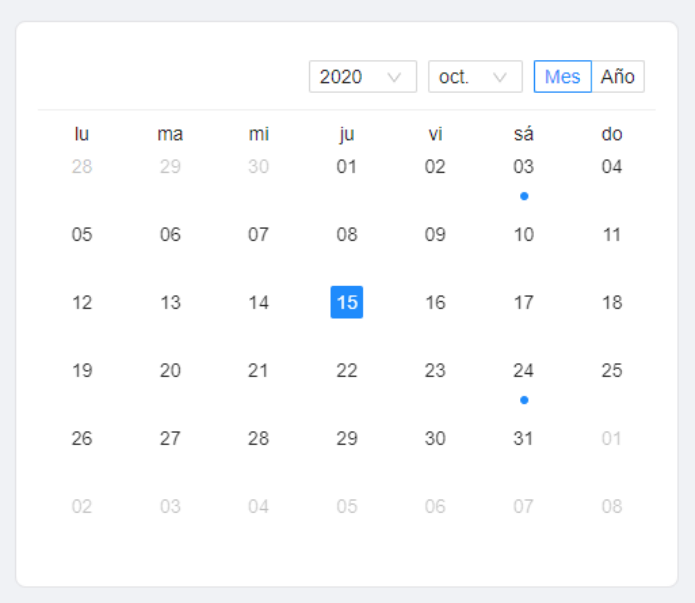


Figura 17 - Componente *Calendar*

- *List:* este componente es el medio por el cual, mediante una lista de eventos obtenida del servidor, se muestra una lista con cada uno de los eventos, junto con sus datos: nombre, fecha, lugar y sociedad a la que pertenece el evento.

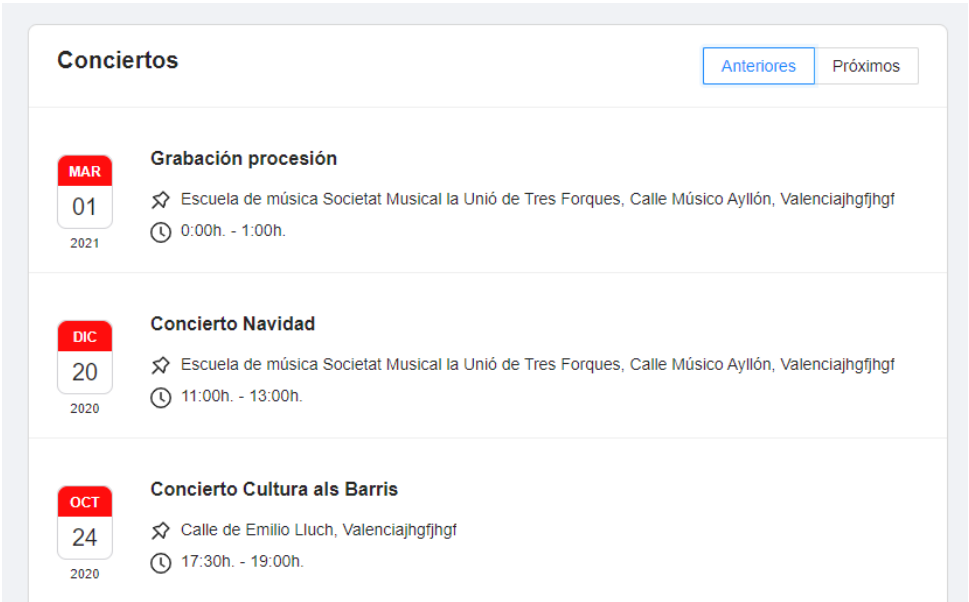


Figura 18 - Componente *List*

- *Form:* es el formulario empleado en toda la aplicación para rellenar campos de texto. Dentro del componente se introducen los subcomponentes Item, que a su vez cada uno contiene un Input (campo de texto), donde se podrá escribir en la interfaz de usuario. El componente Form almacena los valores introducidos en

los campos de texto del formulario y permite su interacción en el contexto de la página del formulario.

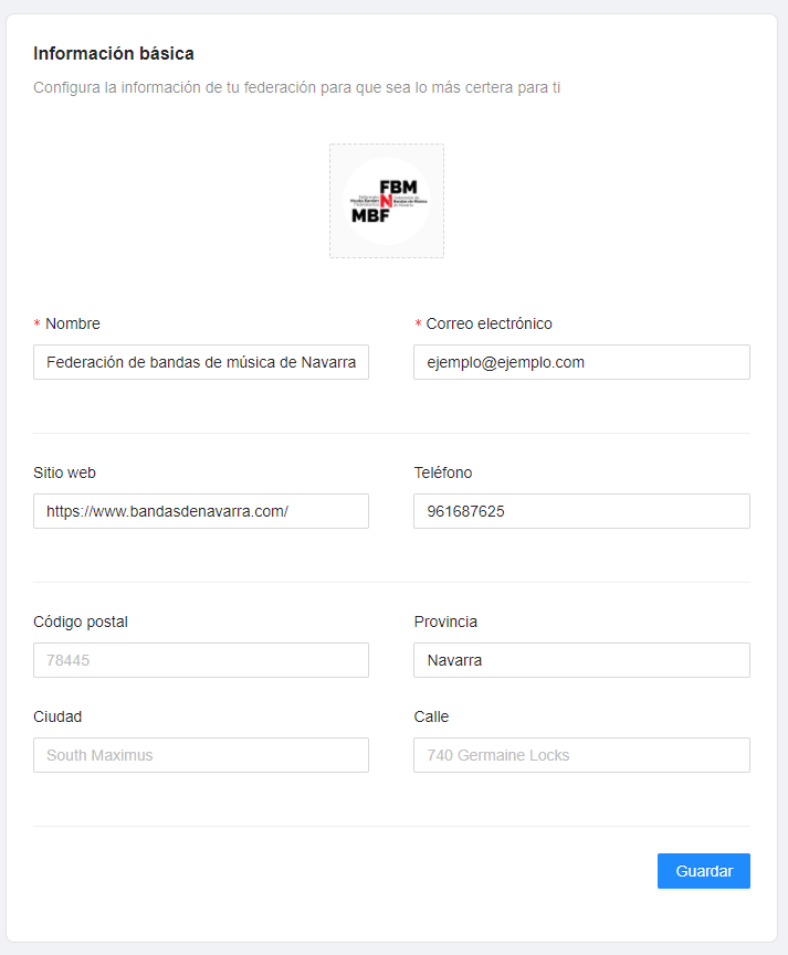


Figura 19 - Componente *Form*

5.1.3.2 Diseño de la interfaz

La interfaz del panel de control está implementada mediante el componente Layout de Ant Design. Este componente estructura la pantalla del panel en varios subcomponentes. El componente Layout trae los estilos predeterminados de Ant Design, pero se han ajustado los estilos para que siga el estilo diseñado para el panel de control de federaciones.

Los subcomponentes del componente Layout son los siguientes:

- *Header*: es la parte superior o cabecera del panel. En la parte derecha del Header podemos encontrar la información del usuario, es decir, su avatar o foto de perfil y, a su derecha, el nombre del usuario y la opción de cerrar sesión.

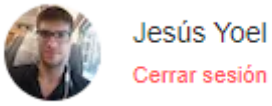


Figura 20 - *Header* de la interfaz

- *Sider*: es la parte lateral del panel, y la principal manera de navegar entre las diferentes páginas del panel de control. En la parte más alta se muestra el logo de la federación autenticada, y justo debajo aparece el nombre de la federación. Inmediatamente después, aparecen los tres enlaces a las diferentes páginas del panel. Respectivamente, son Sociedades, Eventos y Ajustes.

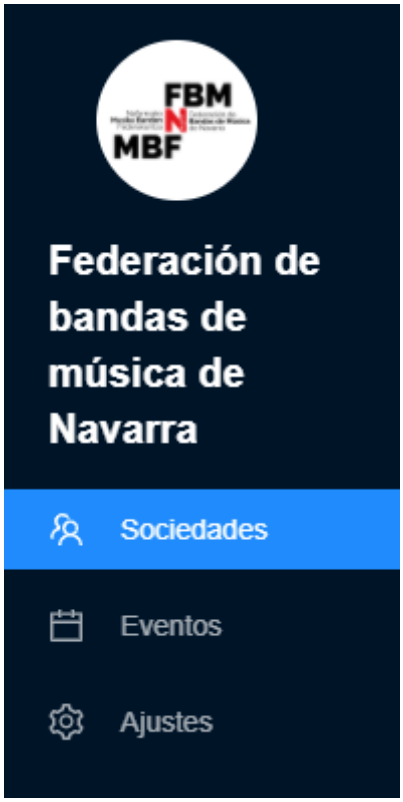


Figura 21 - *Sider* de la interfaz

- *Content*: es la parte contenedora de la página seleccionada. En este subcomponente se mostrará toda la información correspondiente a la página en la que nos encontremos. A diferencia de los dos subcomponentes anteriores,

este puede mostrar diferentes cosas a lo largo del panel, ya que mostrará toda la información correspondiente a la página en la que nos encontremos.

Figura 22 - *Content* de la interfaz de la página Sociedades

La visión conjunta de la interfaz dentro del panel de control y de todos los subcomponentes del Layout de Ant Design sería tal como se puede ver en la Figura 23.

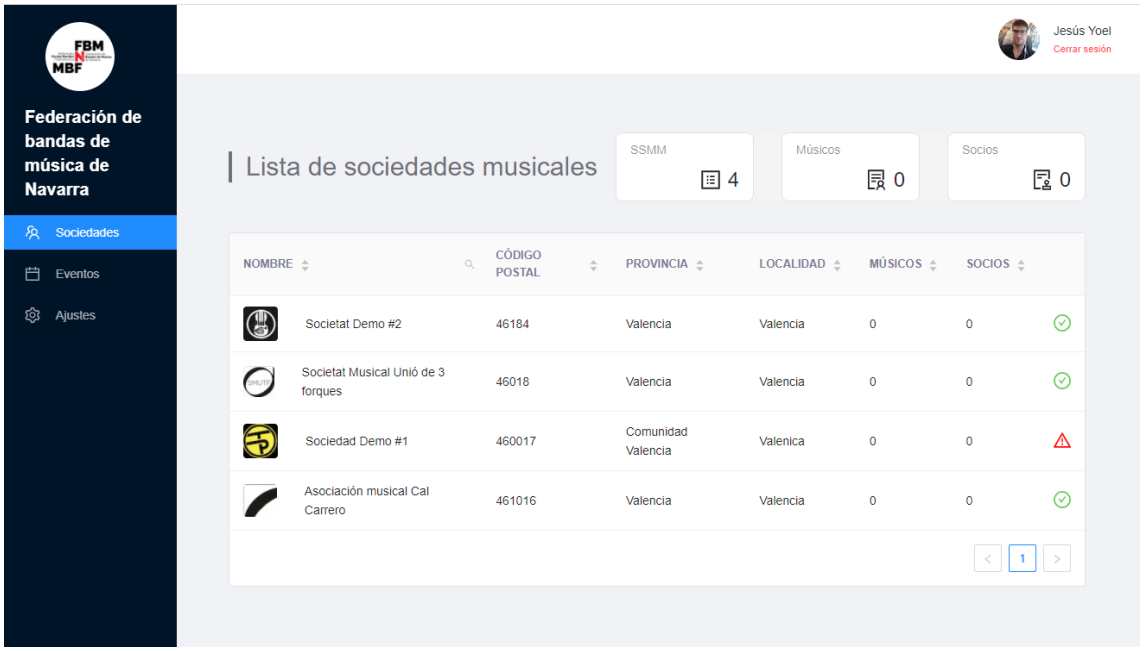


Figura 23 - Interfaz del panel de control

Por otra parte, en otras páginas pertenecientes al proyecto se ha usado una interfaz diferente. La interfaz consiste en una imagen de fondo de músicos y bandas de música. En el centro de la imagen, se ubica un panel con fondo blanco con el contenido de esa página. Todas las páginas que no pertenecen al panel de control contienen esta interfaz. El ejemplo más claro es la página de autenticación.

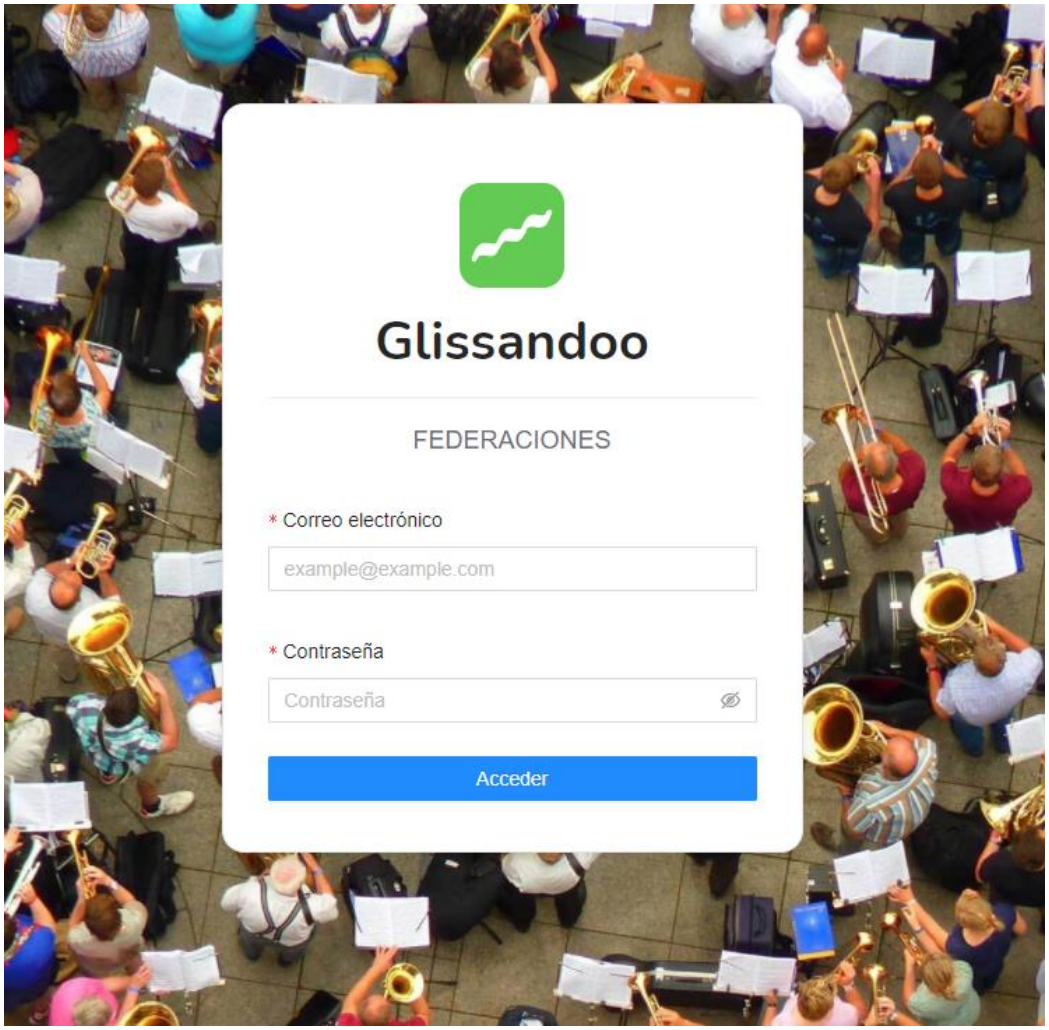


Figura 24 - Interfaz de autenticación

Esta es la estructura que siguen otras páginas como la página de conexión de datos, entre algunas otras.

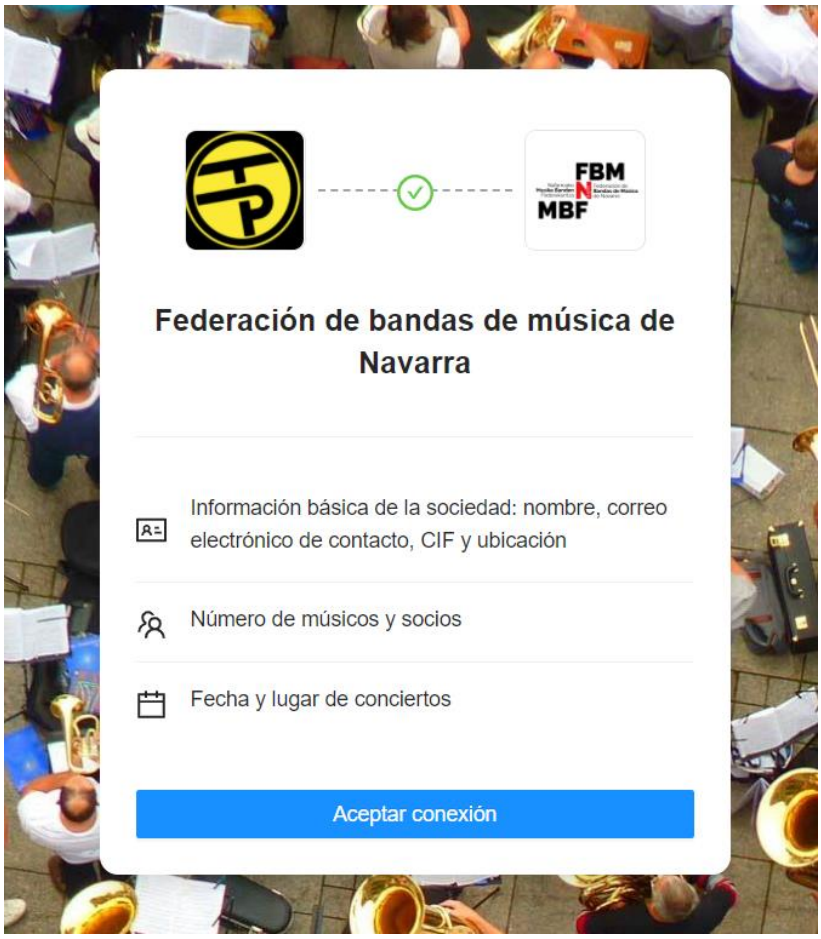


Figura 25 - Interfaz de conexión de datos

5.1.4 Diseño del *back-end*

En estas aplicaciones, desarrolladas en React y NextJS, se usa de forma común una base de datos no relacional (No-SQL), con la información estructurada en un árbol JSON.

Para este proyecto, teniendo en cuenta las necesidades y las dimensiones del mismo, se ha decidido usar Firebase, un servicio de Google que realiza la función de *back-end* y de base de datos.

Sin embargo, debido a las funcionalidades que proporcionaba el plan de pago de Firebase, se ha optado por este plan, ya que ofrecía una utilidad más completa. Se han usado las siguientes funcionalidades:

- Autenticación de Firebase: para permitir la autenticación y asegurar los permisos que requieren el uso del panel de control.
- *Hosting* de *front-end*: para alojar el panel de control bajo el dominio de Glissandoo.
- *Storage*: para almacenar enlaces o archivos usados en el panel de control o subidos por el cliente.
- *Functions*: para almacenar funciones propias de *back-end* y poder ejecutarlas desde el *front-end.*

En el capítulo (2) Estado del arte se ha detallado más información sobre Firebase.

Para realizar este proyecto, se creó un proyecto de Firebase de desarrollo, usando datos de prueba, procurando que fueran lo más semejantes posible a datos reales, de clientes reales. El proyecto está pendiente de una migración al proyecto que usa Glissandoo.

5.1.5 Diseño de la base de datos

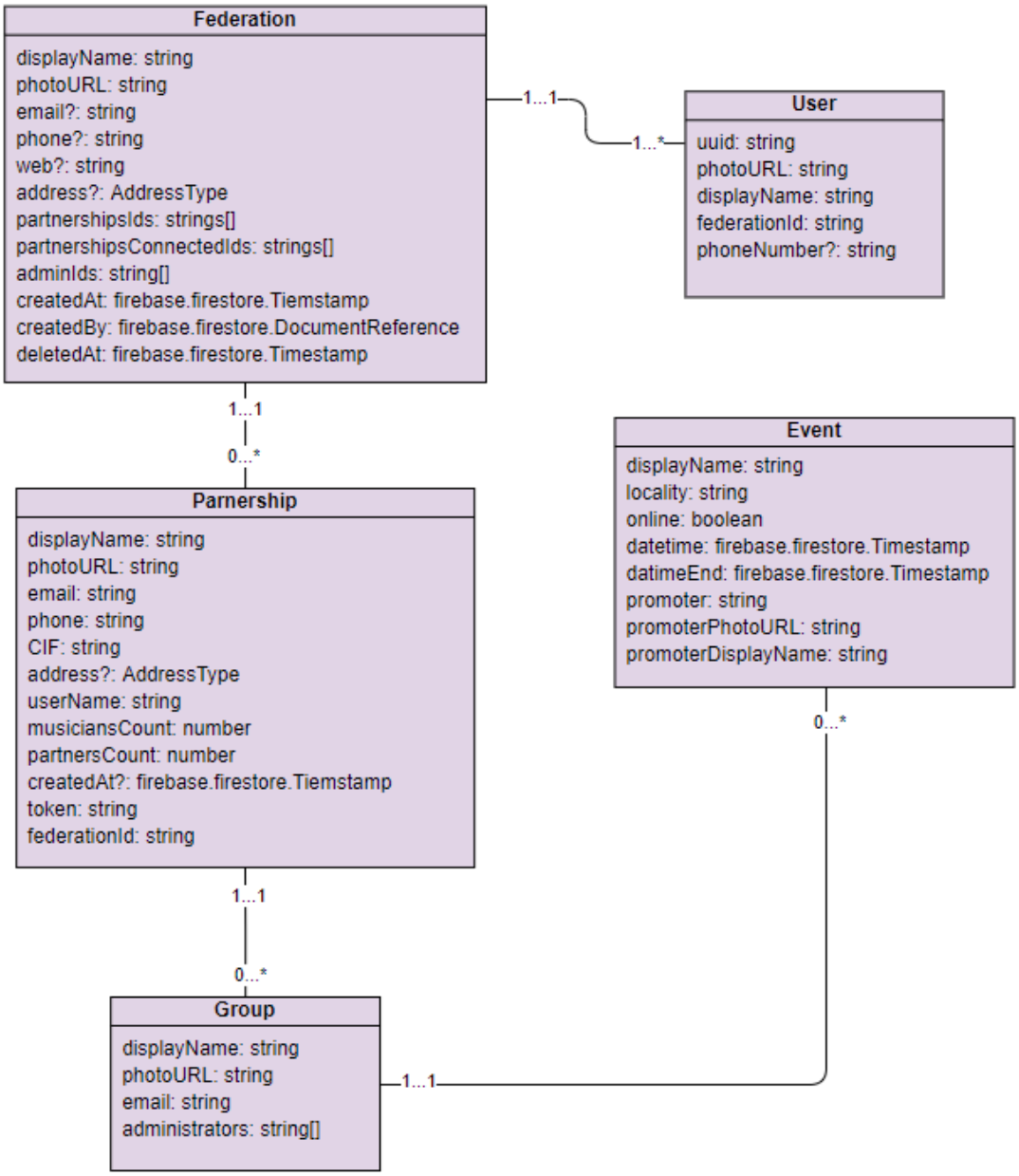
La elección del servicio de Firebase como *back-end* obliga a Glissandoo a emplear su base de datos documental o no relacional. Esto aporta al equipo de desarrollo mayor flexibilidad y comodidad a la hora de manipular los datos en el *front-end*.

Una base de datos no relacional es una base de datos que no utiliza el esquema tabular de filas y columnas que se encuentran en la mayoría de los sistemas de bases de datos tradicionales. En cambio, está constituida por programas que almacenan, recuperan y gestionan datos de documentos estructurados.

En gran parte de los sistemas documentales, las entidades no mantienen relaciones entre ellas que deban reflejarse en el modelo Entidad-Relación. De hecho, en una típica base de datos documental no suele existir ninguna relación entre las entidades representadas que deba ser tenida en cuenta en el modelo Entidad-Relación.

En estos casos, el modelo de Entidad-Relación “únicamente” aporta cierta claridad conceptual y proporciona una terminología común a todos los miembros del equipo que participan en el diseño.

5.1.6 Diagrama Entidad Relación



5.2 Tecnologías utilizadas

Como es habitual, para este tipo de aplicaciones web se diferenciará la estructura en dos capas: *front-end* o capa de presentación del lado del cliente, y *back-end* o capa de acceso a datos como lado del servidor.

5.2.1 *Front-end*

ReactJS fue desarrollado por Facebook como una biblioteca de Javascript *front-end* para construir interfaces de usuario. React utiliza un estilo declarativo de programación para describir el estado de la interfaz de usuario. ReactJS se usa para crear aplicaciones web, así como aplicaciones renderizadas nativas e incluso aplicaciones móviles. ReactJS usa el DOM virtual (Modelo de objetos de documento: permite acceder y cambiar el contenido, el diseño e incluso la estructura del documento).

Por tanto, el cliente de la aplicación está construido bajo el marco de ReactJS, siguiendo las mejores prácticas recomendadas por la documentación disponible de esta librería y su equipo de desarrollo para la creación de la arquitectura de componentes. Esta librería funciona sobre la base de NodeJS, y la acompaña NextJS.

NextJS es un marco de React para desarrollar aplicaciones Javascript de una o más páginas. Una de las principales razones por las que decidió usar NextJS en el proyecto fue el SSR (*Server Side Rendering*). Con él, conseguimos que una vez que se ha entregado el código HTML al navegador del usuario, no sea necesario nada más para que el usuario pueda leer el contenido de la página. Con esto conseguimos que los tiempos de carga de la página parezcan mucho más rápidos para el usuario.

El comportamiento y presentación de la aplicación se basa en las implementaciones en TypeScript compiladas en JavaScript de los componentes de React que implementan la funcionalidad y las vistas usadas en el panel de control.

Los aspectos visuales del panel de control se basan en código HTML de cada componente, que son modificados y cargados dinámicamente según la funcionalidad del mismo. Se usa TailwindCSS para añadir descripciones a los elementos HTML, de manera que se interpreten como código CSS y podamos darle estilos a estos componentes.

Las peticiones a la API REST se realizan usando servicios implementados en la carpeta *helpers* del proyecto, donde cada función hace uso de funciones ubicadas en la sección de *Functions* de la consola de Firebase.

5.2.2 B*ack-end*

Lo más normal en este tipo de aplicaciones web desarrolladas bajo la librería ReactJS y el framework NextJS es hacer uso de APIs desarrolladas en JavaScript. A este tipo de *back-end,* suelen usarse bases de datos no relacionales (No-SQL) y la información suele almacenarse en un árbol estructurado JSON.

Para este proyecto, teniendo en cuenta las necesidades y las dimensiones del mismo, se ha decidido usar Firebase, un servicio de Google que realiza la función de *back-end* y de base de datos.

Sin embargo, debido a las funcionalidades que proporcionaba el plan de pago de Firebase, se ha optado por este plan, ya que ofrecía una utilidad más completa. En el apartado 5.1.4 de este mismo capítulo se ha justificado esta decisión y especificado sus funcionalidades.

5.3 Herramientas usadas

Para la implementación de la aplicación se han utilizado las siguientes herramientas:

- El manejo de control de versiones en un repositorio se ha llevado a cabo mediante GitHub con el Git Bash y la aplicación de escritorio GitHub. La metodología de uso de Git ha sido GitFlow.

- Para la modificación de código, componentes, estilos CSS de elementos del framework de React y NextJS se ha utilizado Visual Studio Code.

- Para el lanzamiento de un servidor local en la versión de desarrollo con ánimo de pruebas y retroalimentación, así como la compilación y descarga y actualización del proyecto se ha utilizado Node.

- Las pruebas y comprobaciones de la web se han realizado en distintos navegadores (Firefox, Edge y Chrome) bajo diferentes resoluciones, para poder realizar un diseño lo más *responsive* posible.
- En cuanto a las modificaciones en la base de datos y configuración de Firebase se ha utilizado la aplicación web “Firebase Console”, con la que se ha podido tener acceso a las funcionalidades de Autenticación, *Firestore Database, Storage, Hosting* y *Functions*.

5.4 Implementación detallada

5.4.1 Implementación de pantalla de *front-end*

Una de las principales funcionalidades del panel de control de federaciones es poder acceder a la lista (en forma de tabla) de sociedades musicales de la federación. Esta es la primera página a la que se accede después de la autenticación.

Figura 26 - Tabla de sociedades musicales

La tabla con las sociedades musicales se ha implementado usando el componente Table de Ant Design, usando cada una de sus Table Columns para mostrar cada uno de los datos más relevantes de cada una de las sociedades.

```
<Table
    rowKey={(item) => item.id}
    onRow={(partnership) => ({
        onClick: () => {
            router.push(`/list/${partnership.id}?filter=>`);
        },
    })}
    className="table-content no-pointer rounded border border-gray-200"
    dataSource={filterItems}
>
        <Table.Column
        title={<FormattedMessage id="settings.address.postalCode" />}
        dataIndex="address"
        sorter={(a: PartnershipListItem, b: PartnershipListItem) => {
            if (a.address && b.address)
                return (
                    parseInt(a.address.postalCode) -
                    parseInt(b.address.postalCode)
                );
        }}
        className="cursor-pointer"
        render={(address: BasicAddressData | null) =>
            address ? address.postalCode : 'N/A'
        }
    />
```

Figura 27 - Componente Ant Design (con una columna)

Otra de las funcionalidades principales del panel de control es consultar los detalles de cada sociedad. Esto es posible debido a que cada columna de la tabla de Ant Design contiene un enlace a la página de los detalles de su respectiva sociedad musical. El contenido de esta página variará, dependiendo de que las sociedades musicales hayan dado su consentimiento y hayan cedido sus datos a su correspondiente federación y a Glissandoo.

Figura 28 - Detalles de la sociedad conectada

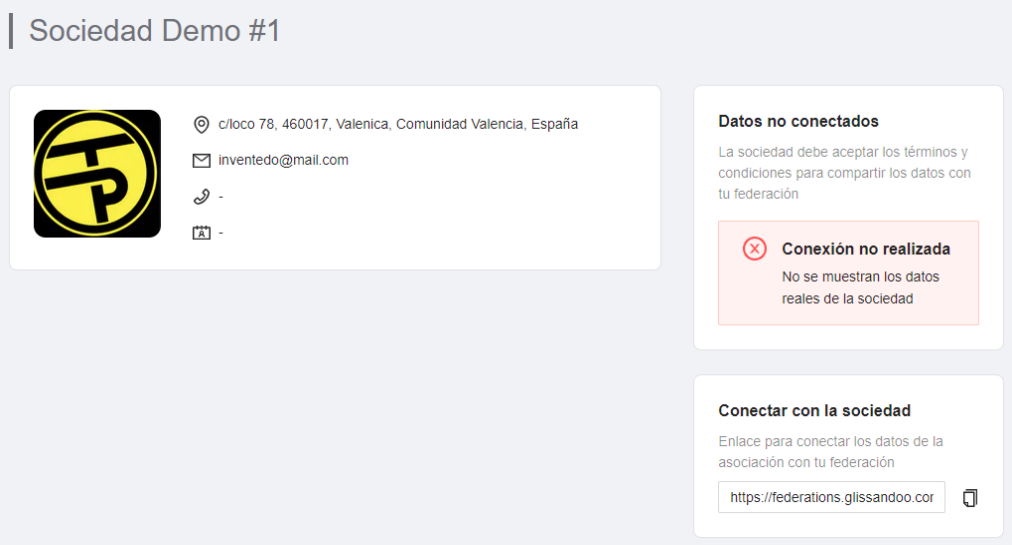


Figura 29 - Detalles de la sociedad no conectada

Para saber si la página debe cargar o no algunos datos, como las estadísticas y los eventos, se consulta el atributo *connectedAt* de la sociedad, que será nulo si la sociedad no ha conectado los datos, o una marca de tiempo (fecha y hora) del momento cuando conectó los datos.

Por otra parte, otra de las funcionalidades principales del panel de control son los ajustes de la federación, es decir, modificar los datos correspondientes a la federación a la que pertenece el usuario autenticado (normalmente, líder de la

federación). Entre los datos que se pueden modificar están el nombre de la federación y el correo electrónico como campos obligatorios, y la página web, el teléfono, la dirección (código postal, provincia, ciudad y calle) y la imagen de la federación, la cual se puede cambiar por un archivo local recortable para la adecuación del tamaño del logo en el panel de control.

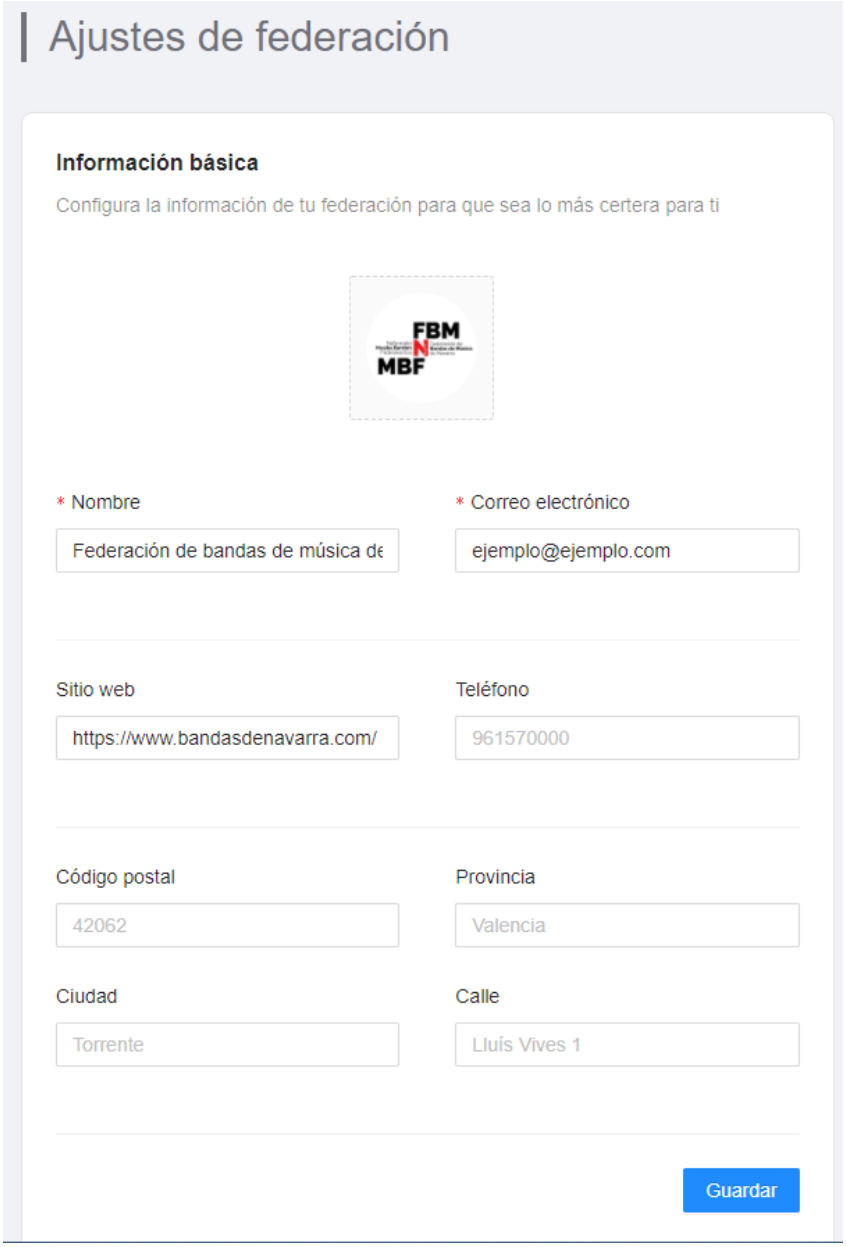


Figura 30 - Ajustes de la federación

Para desarrollar esta vista se ha usado el componente Form de Ant Design para rellenar los campos de texto, junto con el componente Upload para modificar el logo de la federación.

```
<Form
    name="nest-messages"
    layout="vertical"
    className="w-100"
    onFinish={onFinish}
    validateMessages={validateMessages}
    initialValues={{
        displayName: federation.displayName,
        email: federationSettings.email,
        web: federationSettings.web ?? '',
        phone: federationSettings.phone ?? '',
        address: federationSettings.address,
    }}
>
    <Row gutter={40}>
        <Col span={12}>
            <Form.Item
                name={'displayName'}
                label={formatMessage({
                    id: 'auth.name.placeholder',
                })}
                rules={[
                    {
                        required: true,
                        message: formatMessage({
                            id: 'form.required',
                        }),
                    },
                ]}
            >
                <Input />
            </Form.Item>
        </Col>
```

Figura 31 - Componente Form de Ant Design

```
<Upload
    name="avatar"
    listType="picture-card"
    className="avatar-uploader"
    showUploadList={false}
    onChange={async (event) => {
        setVisible(true);
        const aBuffer =
            await event.file.originFileObj.arrayBuffer();
        setArrayBuffer(aBuffer);
        setImage(URL.createObjectURL(event.file.originFileObj));
    }}
    accept="image/png, image/jpeg"
>
    <Avatar
        size={{
            xl: 80,
            md: 80,
        }}
        src={
            <img
                className="rounded-xl"
                src={image}
                alt="Logo de la federación"
                width="100%"
            />
        }
    />
</Upload>
```

Figura 32 - Componente Upload de Ant Design

5.4.2 Implementación de la conexión a base de datos

La conexión entre el *Back-end as a Service* y la aplicación React es bastante simple. Podemos servirnos de la API de Firebase mediante React, que maneja las llamadas a los servicios. Para ello, es necesario instalar el paquete e incluirlo en el módulo de dependencias de la aplicación:

```
import firebaseAdmin from 'firebase-admin';

const settings = {
    serviceAccount: require('../serviceAccountKeyDev.json'),
    databaseURL: 'https://glissandoo-dev-78ed4.firebaseio.com',
};

if (!firebaseAdmin.apps.length) {
    firebaseAdmin.initializeApp({
        credential: firebaseAdmin.credential.cert(settings.serviceAccount),
        databaseURL: settings.databaseURL,
    });
}

export { firebaseAdmin };
```

Figura 33 - Conexión a Firebase

La configuración de Firebase en el entorno de la aplicación tiene que corresponder con la del proyecto en la consola de Firebase. Toda esa información está contenida en el archivo *serviceAccountKeyDev.json*, que contiene las claves para la conexión con la consola.

Una vez implementados los servicios, estas interacciones entre Firebase y React se hacen muy intuitivas, ya que podemos simplemente llamar a los métodos de Firebase para nuestros intereses.

```
export const getFederation = async (id: string) => {
    const doc = await firebaseAdmin
        .firestore()
        .collection(CollectionNames.Federation)
        .doc(id)
        .get();
    const federation = new Federation(doc);
    if (!federation.isActive) {
        throw new Error('This federation is not valid');
    }
    return federation;
};
```

Figura 34 - Ejemplo de función proporcionada por Firebase

5.4.3 Implementación de la autenticación de Firebase

Firebase ofrece la funcionalidad Firebase Auth, la cual nos permite registrar, iniciar sesión y cerrar sesión usando correo electrónico, contraseña, teléfono, etc. En este proyecto se ha llevado a cabo la autenticación de usuario mediante correo electrónico y contraseña. Aunque el panel de control no permite el registro de usuarios, ya que los usuarios son registrados por el cuerpo técnico de Glissandoo con sus correspondientes credenciales, sí permite iniciar y cerrar sesión. Tanto a la hora de iniciar sesión como de cerrar, se usan las funcionalidades pertinentes proporcionadas por Firebase Auth, con la diferencia que el inicio de sesión usa como parámetros el email y la contraseña introducidas por el usuario.

```
async function signOut() {
    try {
        await firebase.auth().signOut();
        router.push('/');
    } catch (err) {
        notification.error({
            message: formatMessage({ id: 'home.contact.error' }),
        });
    }
}
```

Figura 35 - Función de cerrar sesión

Sin embargo, en el proceso de comprobación de credenciales y autenticación en el inicio de sesión se hace una operación extra. Se llama, usando el usuario con la sesión ya iniciada, a Firebase Firestore y obtenemos el usuario almacenado que se corresponda con el de la sesión ya iniciada. Esto se realiza única y exclusivamente para comprobar que existe un usuario en Firestore que coincide con el usuario proporcionado por el servicio Auth, y que además este usuario contenga el atributo *federationId.* Si alguna de estas comprobaciones no llega a realizarse correctamente, al usuario no se le permitirá el acceso al panel de control, pues no es un usuario con los permisos necesarios para acceder, es decir, que sea líder o administrador de la federación.

```
const onSignin = async (values: FormValues) => {
    setLoading(true);
    const { email, password } = values;
    try {
        const userCredential = await firebase
            .auth()
            .setPersistence(firebase.auth.Auth.Persistence.SESSION)
            .then(async () => {
                return await firebase
                    .auth()
                    .signInWithEmailAndPassword(email, password);
            });

        const userDoc = await firebase
            .firestore()
            .collection(CollectionNames.User)
            .doc(userCredential.user.uid)
            .get();
        const user = new User(
            userDoc as unknown as FirebaseFirestore.DocumentSnapshot
        );

        if (!user.federationId) {
            notification.error({
                message: formatMessage({
                    id: 'signin.noFederation',
                }),
            });
            setLoading(false);
            return;
        }

        setLoading(false);
        router.push(`/list`);
    } catch (err) {
        notification.error({
            message: formatMessage({ id: err.code }),
        });
```

Figura 36 - Función de iniciar sesión

5.5 Testing

Hacia el final del proyecto, se ha ido implementando una subsección de testing en la que se comprueba que las funciones que se comunicarán con la base de datos y funciones de lógica relevante funcionan correctamente.

Debido a que esta fase del proyecto se realizó justo al final del periodo de prácticas del alumno en la empresa Glissandoo, no se ha podido realizar un testing completo de la aplicación y sus componentes, por lo que quedará como asignatura pendiente en un futuro. De hecho, ha sido la propia empresa la que ha decidido que este testing más completo se hará en un futuro.

El objetivo de este testing es automatizar pruebas que comprueben el correcto funcionamiento de todos los componentes y funciones probados, de forma que en el

futuro sea más seguro realizar actualizaciones, refactorizaciones, añadir funcionalidades o cambiar el comportamiento actual del panel de control. Estos test ayudarán a comprobar que, después de realizar cualquier futuro cambio, en caso de que este cambio lleve al fallo o incorrecto funcionamiento de otros componentes anteriormente funcionales, ubicar el fallo rápidamente, y saber qué ha fallado exactamente, lo cual será una guía muy valiosa para solventar el problema.

En definitiva, este último esfuerzo y dedicación contribuirá a mejorar la mantenibilidad futura del panel de control y abaratar costes en cuestión de refactorización y actualizaciones.

6. Resultados, conclusiones y futuro

En este proyecto se ha tratado de abordar la necesidad de la empresa de facilitar el acceso a los datos de las sociedades y de la propia federación por parte de los administradores de las federaciones. También ofrece la posibilidad de conectar las sociedades a las federaciones, de forma que se quede registrada la presencia de una sociedad musical en su federación.

El uso de buenas prácticas y las tecnologías y *frameworks* de desarrollo web mencionados anteriormente se han mantenido como un aspecto vital durante todas las fases del proyecto. No solo se busca la calidad del producto final, también se busca la comodidad a la hora de desarrollar el software y la experiencia ganada en estas prácticas.

El papel de control se implementó mayoritariamente en horario de prácticas externas en la empresa Glissandoo bajo la supervisión del CEO y CTO de la empresa. Parte de la funcionalidad y, en sobremanera, las tecnologías a usar fueron decisión del startup.

Gracias a que anteriormente había estado en contacto con Javascript y ReactJS me fue posible amoldarme con facilidad a las tecnologías que me propuso el equipo para llevar a cabo el proyecto. Se me asignaron dos proyectos. El primero fue la replicación de la página web de Glissandoo y el segundo fue la creación del panel de control federaciones, ambas con las mismas tecnologías.

El producto final cumple con todos los objetivos propuestos desde el inicio del proyecto, añadiendo algunas funcionalidades que no estaban previstas, como por ejemplo la sección de eventos de la federación. Gracias a la aplicación de estas tecnologías tan punteras, el panel de control cuenta con una robustez y arquitectura confiables.

Se ha usado ReactJS y NextJS con una arquitectura desarrollada en TypeScript para el *front-end* y Firebase, un "*Back-end as a Service*" que ha resultado en una válida e interesante elección para proyectos de tamaño medio.

El resultado es una aplicación web con una experiencia de usuario propia de aplicaciones web actuales, especialmente para ordenadores de escritorio.

6.1 Posibles ampliaciones

A nivel más personal, estos proyectos han tenido un gran valor en lo que al aprendizaje de nuevas tecnologías y desarrollo tanto de *front-end* como *back-end* se refiere. Sin embargo, estos meses de prácticas no han sido un mero ejercicio o prueba académica. La idea es que este proyecto derive en un producto con usabilidad real, y por tanto que pueda aportar beneficios al startup Glissandoo.

Es por esto que el proyecto será usado y, probablemente, ampliado para las necesidades de los administradores de las federaciones musicales.

6.2 Cambio de *back-end* a uno propio

Este cambio sería posible considerarlo si hubiera un crecimiento significativo en la aplicación y en su tráfico de datos, o bien debido a cambios en la funcionalidad que no puedan ser satisfechos por la flexibilidad del *back-end* actual.

6.3 Otros usos futuros

Además del uso por parte de las federaciones de sociedades, es posible que se extienda el uso del panel de control especificado en este proyecto por organismos con autoridad superior a las anteriores, como la mencionada en el apartado 2 CESM (Confederación Española de Sociedades Musicales). Por ejemplo, la idea a largo plazo de este panel de control es que pueda llegar a ser una herramienta usada por un miembro del senado, para supervisar la actividad de las federaciones musicales. Este miembro podrá aportar en sus funciones oficiales datos y estadísticas de las federaciones al senado, estadísticas y números que podrá consultar en el panel de control de Federaciones de Glissandoo.

7. Referencias bibliográficas

[1] Steven G. Blank, “The four steps to the epiphany”

[2] COESSM - CONFEDERACIÓN ESPAÑOLA SOCIEDADES MUSICALES

https://coessm.org/confederacion/fines-de-la-cesm/

[3] Chorus Connection - About Choir Management Software, https://www.chorusconnection.com/

[4] David Flanagan, O'Reilly Media Inc, “JavaScript: The Definitive Guide”, 7th Edition, pp. 38 - 60, 2020

[5] Dan Vanderkam, O'Reilly Media Inc, “Effective TypeScript: 62 Specific Ways to Improve Your TypeScript”, pp. 1- 20, 2019

[6] Danny Goodman, O'Reilly Media Inc., “Dynamic HTML: The Definitive Reference”, pp. 167 - 171, 2007

[7] Jon Duckett, John Wiley & Sons, Inc., “HTML & CSS Design and Build Websites”, pp 236 - 405, 2011

[8] Alex Banks, Eve Porcello, O'Reilly Media Inc, “Learning React: Modern Patterns for Developing React Apps” 2nd Edition, pp 32 - 61, 2020

[9] Kirill Konshin, “Next.js Quick Start Guide”, pp. 5 - 8, 2018

[10] Matt Frisbie, “Angular 2 Cookbook”, pp. 7 - 13, 2017

[11] Callum Macrae, “Vue.js: Up and Running”, pp. 1 - 20, 2018

[12] Firebase - Documentación, https://firebase.google.com/docs?hl=es-419

8. Anexo

8.1 Startup

Como se puede observar en la Figura 1, hay cuatro factores que construyen esta metodología. [1]

El primer factor se llama *Customer Discovery* y se trata de testear una serie de hipótesis antes de lanzarnos a desarrollar nada. Lo que realmente se busca es saber si realmente se ha encontrado un problema que un número tanto de personas como de empresas o instituciones necesitan resolver, y si la solución propuesta es una solución válida para ellos. En este paso se suele usar la filosofía MVP, Producto Mínimo Viable (*Minimum Viable Product*), para poder aprender de los posibles clientes (*early adopters*) siempre respecto a nuestras hipótesis.

El segundo factor se llama *Customer Validation* y se centra en crear un plan de ventas que sea repetible y que esté enfocado en los *early adopters*, para asegurarnos de que realmente podemos vender nuestro producto como solución a su problema. En general, si los dos primeros factores son satisfactorios, es muy posible que el modelo de negocio previsto sea válido.

El tercer factor se llama *Customer Creation* y trata de generar demanda del producto para alimentar a los canales de ventas enfocados por parte del startup. Se da el salto al *mass market*, por lo que comienzan a aparecer gastos de marketing y ventas considerables.

El cuarto y último factor es *Company Building*, que consiste en afianzar la estructura empresarial del startup cambiando el caos típico de los startups por una estructura en departamentos orientados ya no respecto al cliente sino a su propia misión.

8.2 *Extreme Programming*

Está diseñada para ofrecer el software que los usuarios necesitan en el momento adecuado. En este sentido, ayuda a los desarrolladores a ajustarse a los requerimientos cambiantes de los clientes.

Este tipo de programación se diferencia de las metodologías tradicionales en que pone más énfasis en la adaptabilidad que en la previsibilidad. El *Extreme Programming* considera que los cambios de requisitos sobre la marcha son acciones naturales e inevitables en el desarrollo de un proyecto.

Por ello, un factor clave del *Extreme Programming* es la comunicación, y este es un factor clave porque se trata sobre todo de la comunicación (o *feedback*) por parte del cliente.

Gracias a las presentaciones de los resultados del producto cada poco tiempo, se minimiza el riesgo de tener que rehacer partes que no cumplen con las expectativas de los clientes. También, por otro lado, ayuda a los programadores a centrarse en las tareas más importantes.

8.3 Kanban

Kanban es una palabra japonesa que se refiere a tarjetas visuales. Esta palabra se compone de Kan y Ban. Kan significa visual y Ban significa tarjeta.

Como método de visualización, puede aclarar el estado del proyecto de un vistazo y asignar nuevas tareas de una manera muy eficaz. Para aplicarlo es necesitario un tablero de tareas, puedes usarlo para mejorar tu trabajo y tener un ritmo sostenible.

Dispone de un panel que todos los miembros del equipo puedan ver y acceder. El estado del flujo de tareas se registra en la columna, y cada tarea irá moviéndose entre columnas de principio a fin según el estado en el que se encuentren.

Es un tablero continuo en el que las cartas no se mueven, pero a medida que la gente avance por él, se irán acumulando nuevas funcionalidades, mejoras o eventos al principio. Por tanto, se pueden priorizar y colocar cada tarea en la parte más adecuada.

8.4 SCRUM

SCRUM es un proceso en el que se aplican periódicamente un conjunto de buenas prácticas al trabajo en equipo y se obtienen los mejores resultados del proyecto. Las entregas parciales y regulares del producto final se organizan de acuerdo con los beneficios que aportan al destinatario del proyecto.

Por este motivo, es especialmente adecuado para proyectos en entornos complejos, donde los resultados deben obtenerse lo antes posible y los requisitos cambian constantemente o están poco definidos. La innovación, la competitividad, la flexibilidad y la productividad son fundamentales.